

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



### THESIS

K1422

**PILOT EMERGENCY TUTORING SYSTEM  
FOR F-4 AIRCRAFT FUEL SYSTEM MALFUNCTION  
USING MEANS-ENDS ANALYSIS**

by

Moung-Hung Kang

June 1990

Thesis Advisor:

Neil C. Rowe

Approved for public release; distribution is unlimited.

T252351

20100915191

## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable) 52	7b. ADDRESS (City, State, and ZIP Code)	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)	10. SOURCE OF FUNDING NUMBERS	
8c. ADDRESS (City, State, and ZIP Code)		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) PILOT EMERGENCY TUTORING SYSTEM FOR F-4 AIRCRAFT FUEL SYSTEM MALFUNCTION USING MEANS-ENDS ANALYSIS			
12. PERSONAL AUTHOR(S) Moung-Hung, Kang			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day) June 1990	15. PAGE COUNT 137
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Artificial Intelligence, Intelligent Tutoring System, Pilot Associate	
	SUB-GROUP	Means-ends Analysis, Expert System	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This research studies making an intelligent tutoring system to reduce pilot mistakes during an inflight emergency, specifically the flight environment for an F-4 aircraft fuel system malfunction. The pilot emergency tutoring system consists of an expert system with knowledge of the F-4 aircraft fuel system and a tutoring system with strategy to teach students. The expert and tutoring systems use a means-ends analysis problem approach. The means-ends analysis approach reduces differences between the current state and goal state until the final state or an unsolvable state is reached. Through use of the program, pilots can learn to prevent decision-making errors and procedural errors. Consequently, the pilot can be freer to serve as aircraft commander during an emergency situation.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof Neil C. Rowe		22b. TELEPHONE (Include Area Code) (408) 646-2168	22c. OFFICE SYMBOL 52

Approved for public release; distribution is unlimited.

**PILOT EMERGENCY TUTORING SYSTEM  
FOR F-4 AIRCRAFT FUEL SYSTEM MALFUNCTION  
USING MEANS-ENDS ANALYSIS**

by

Moung-Hung Kang  
Major, Korean Air Force  
B.S., Korean Air Force Academy, 1981

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the


**NAVAL POSTGRADUATE SCHOOL**  
June 1990

Authors:

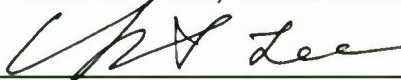


Moung-Hung Kang

Approved By:



Neil C. Rowe, Thesis Advisor



Yuh-jeng Lee, Second Reader



For Robert B. McGhee, Chairman,  
Department of Computer Science

## ABSTRACT

This research studies making an intelligent tutoring system in order to reduce pilot mistakes during inflight emergency, specifically the flight environment within an F-4 aircraft fuel system malfunction. The pilot emergency tutoring system consists of an expert system with knowledge of the F-4 aircraft fuel system and a tutoring system with strategy to teach students. The expert and tutoring systems use a means-ends analysis problem approach. The means-ends analysis approach reduces differences between the current state and goal state until the final state or an unsolvable state is reached. Through use of the program, pilots can learn to prevent decision-making errors and procedural errors. Consequently, the pilot can be freer to serve as aircraft commander during an emergency situation.



140015  
K1422  
C.1

## TABLE OF CONTENTS

I.	INTRODUCTION .....	1
A.	BACKGROUND .....	1
B.	NEED FOR A PILOT TUTORING SYSTEM.....	2
C.	IDEA OF PROBLEM SOLVING .....	3
D.	THESIS ORGANIZATION.....	3
II.	SUMMARY OF PREVIOUS WORK .....	5
A.	SURVEY OF ARTIFICIAL INTELLIGENCE IN AVIONICS .....	5
1.	The Systran Corporation Expert System Pilot Aid .....	5
2.	Texas Instruments Emergency Procedures Expert System.....	6
3.	McDonnell Douglas Avionics Integrated Maintenance Expert System .....	6
B.	BACKGROUND OF THE TUTORING SYSTEM .....	6
1.	Means-ends Analysis .....	7
2.	Means-ends Tutor .....	7
III.	CONCEPT OF THE PILOT TUTORING SYSTEM.....	9
A.	DESCRIPTION OF F-4 AIRCRAFT AND FUEL SYSTEM.....	9
1.	F-4 Aircraft .....	9
2.	F-4 Fuel System .....	9
B.	PILOT PROCEDURE .....	10
1.	Normal Procedure .....	10
2.	Emergency Procedure .....	11
C.	OBJECTIVE .....	11
D.	ASSUMPTIONS.....	12
IV.	IMPLEMENTATION OF THE PILOT TUTORING SYSTEM .....	13
A.	PROGRAMMING LANGUAGE.....	13

B.	ORGANIZATION OF THE PROGRAM .....	13
1.	Student Model .....	13
2.	Problem Model .....	14
3.	Expert Domain Model.....	14
4.	Tutor Domain Model .....	15
C.	DATA STRUCTURES IN PROGRAM.....	15
1.	Input and Output .....	15
2.	World Knowledge .....	15
a.	Definition of Switch and Indicator Position .....	16
b.	Means-ends Analysis .....	17
3.	Tutoring System.....	18
a.	Tutor.....	18
b.	Means_ends_tutor .....	18
c.	Check_with_student.....	18
d.	Handle_student_op .....	18
e.	Writelist .....	18
f.	Once_means_ends.....	18
D.	IMPLEMENTATION OF THE EMERGENCY PROCEDURE .....	19
1.	Simplification of the Global Variable.....	19
2.	Implementation of Means-ends Analysis in Emergency Procedure .....	19
a.	Reconmended Operator .....	19
b.	Precondition of Operator.....	20
c.	Deletepostcondition of Operator.....	21
d.	Addpostcondition of Operator .....	21
E.	IMPLEMENTATION OF THE TUTOR.....	22
1.	Student Execution Error.....	22
2.	Natural Language.....	23

V.	ANALYSIS.....	24
VI.	CONCLUSIONS .....	27
A.	CONCLUSIONS .....	27
B.	FURTHER WORK .....	27
	APPENDIX A TEST OF EMERGENCY CASE .....	28
A.	INTERNAL WING FUEL FAILS TO TRANSFER.....	28
B.	CENTERLINE FUEL FAILS TO TRANSFER.....	34
C.	EXTERNAL WING FUEL FAILS TO TRANSFER.....	40
D.	REVERSE TRANSFER OF FUSELAGE FUEL.....	46
E.	FUEL STREAMING FROM UNDERSIDE OF AIRCRAFT FOLLOWING CENTER JETTISON.....	53
F.	WING FUEL LEAK .....	58
G.	NORMAL SITUATION .....	63
H.	GENERAL CASE .....	68
I.	RANDOM SELECTING CASE.....	78
	APPENDIX B SOURCE CODE OF EXPERT SYSTEM .....	86
	APPENDIX C SOURCE CODE OF TUTORING SYSTEM .....	107
	LIST OF REFERENCES .....	125
	INITIAL DISTRIBUTION LIST .....	127

## LIST OF TABLES

TABLE I	RESULT OF OUTPUT .....	26
---------	------------------------	----



## ACKNOWLEDGMENTS

Thanks god for my study in United States. And I'd like thank Korean Air Force for providing this educational opportunity.

I'd like thank Professor Neil C. Rowe and Yuh-jeng Lee, whose patience and guidance allowed this work to be completed.

I also extend my thanks to Mrs. Ann Warner who edited the initial draft of thesis. Finally, I wish to thank my lovely wife Eunju and children Keon for their patience of United States life.

## **I. INTRODUCTION**

### **A. BACKGROUND**

During the past 20 years, the study of Artificial Intelligence (AI) in machines has evolved from a purely academic activity to a major growth industry involved in both government and commercial applications. In particular, in the last five years, there has been an explosive growth in the number of AI systems capable of giving practical advice on a variety of problems ranging from automechanics to corporate-level policy decision-making. [Ref. 1]

During the same period, AI technology-related activity within the military has increased dramatically. This heightened interest and expanding investment in AI by the Department of Defense (DOD) and individual services (Army, Navy, Air Force, and Marine Corps) may be attributed to a number of factors:

1. The movement of AI technology from academic research to real world application.
2. The significant advances in the speed and accuracy of sensors and weapons through the increasing complexity of modern-day military operations.
3. A growing awareness and acceptance by the military of the potential of AI technologies to help solve military problems. [Ref. 2]

One major drive behind the interest is the funding and research base provided by the Defense Advanced Research Projects Agency's Strategic Computing Initiative, particularly in areas targeted by DARPA for initial technology application: an Autonomous Land Vehicle; an Intelligent Pilot's Associate; and Naval Battle Management. [Ref. 3]

Although there has been real progress in AI techniques in the aeronautic industry, there will never be absolute freedom from aircraft accidents. Recent news of big aircraft accidents has gradually increased concern for airline safety.

## **B. NEED FOR A PILOT TUTORING SYSTEM**

The flight phase can be divided into: taxiing, takeoff, climb, cruise or mission, descent and landing phases. All except the taxiing phase can be included in the inflight phase. This study will focus on the inflight phase, since grounded aircraft pose no dangerous situations.

During an inflight emergency the pilot has to react immediately to solve the problem. No help can be expected from other aircraft, and pilot action is not recoverable. Psychological conditions make mistakes more likely. During an emergency, the pilot receives a great deal of information and advice, but it is not always helpful. If the pilot wants specific advice concerning the problem, he has to explain the current status of every component, something that is impossible during an emergency.

The major factors behind aircraft accidents consist of environmental extremes, material failure and human error [Ref. 4]. The procedures and judgmental errors of air traffic controllers, flight commanders, and flight supervisors will be included in human error. Aviation Week and Space Technology [Ref. 5] reported "The human factor is almost 65% in jet transport accidents." Pilot mistakes account for the majority of human error. It is difficult for the pilot, during the emergency, to make a decision based on the indicators and gages available.

An emergency expert system or an emergency tutoring system can help pilots to reduce pilot error. An emergency expert system needs sensors or gages for every component to provide input data. The expense may be prohibitive. A tutoring system which has the same knowledge base as such an expert system requires no additional hardware, though its efficiency is less than that of the expert system.

Modern aircraft such as the F-16 and F-18 still use conventional teaching methods in flight training courses. Bloom stated a definition and an efficiency in an intelligent tutoring system:

Conventional teaching, which means a teacher presenting material in front of 20-200 people, is one of the least effective methods for educational delivery. The results achieved through conventional teaching were in the 50-60 median range.

However, and here as the important part of this study, students involved in one-to-one tutoring seem to perform around 98th percentile as compared with traditionally trained students. These results were reproduced four times with three different age groups on two different subjects. This study provides evidence that tutoring is one of the most effective educational delivery methods. [Ref. 6]

### **C. IDEA OF PROBLEM SOLVING**

Our Pilot Emergency Tutoring System (PETS) consists of expert and tutor domains. The expert domain contains data of the aircraft systems as well as solutions to emergency situations. The tutor domain has computer-based ideas to solve the current problem and provides strategies to teach students how to correct their mistakes.

The problem approach uses the means-ends analysis problem-solving technique in both domains. Means-ends analysis is useful in both realistic problems and those that can be stepped through. Previous work in this area was done by Porter [Ref. 7] who made a decision support system in the AH-1T engine failure emergency. This study uses Rowe's [Ref. 8:p. 263] means-ends analysis structure. Specifically, it uses conditional fact clauses instead of production rules to decide diagnostic problems.

### **D. THESIS ORGANIZATION**

Chapter II will study the area of artificial intelligence in avionics and explain the basic idea of the means-ends analysis technique and means-ends tutor. The concept



of pilot procedures during normal and emergency situations will be presented in Chapter III. Chapter IV will discuss the hardware environment, data structure, system model, and detailed application of the pilot procedure. Chapter V will analyze the work and present conclusions with explanations of further work.

## II. SUMMARY OF PREVIOUS WORK

### A. SURVEY OF ARTIFICIAL INTELLIGENCE IN AVIONICS

AI research and exploratory development has recently made real progress in avionics. Specifically, the development has concentrated on the pilot's associate and maintenance and trouble-shooting of equipment but areas such as mission planning, automatic test equipment (ATE) and built-in test (BIT) have used AI in real applications.

The combat pilot is already overworked, stressed by requirements and too busy to plan. He must maintain flight control, conduct continuous threat analysis, appropriate countermeasures and monitor and correct aircraft navigation. [Ref. 9:p. 89]

From the pilot's point of view, it is difficult to make a correct decision in every situation; an increased level of automation that provides an integrated avionics system is needed. Additionally, the pilot must be considered an aircraft commander, with a situation-assessment manager, a tactical-planning manager, a mission-planning manager, and a systems-status manager under his control.

#### *1. The Systran Corporation Expert System Pilot Aid*

The Systran Corp. developed the Expert System Pilot Aid (ESPA) with the U.S Air Force's Avionics Laboratory to study the application of expert system programming techniques to the advanced fighter cockpit. They selected three scenarios: engine failure, landing gear failure and systems failure on takeoff in an F-16. During these emergencies, the pilot can correct the problem by evoking ESPA with a HELP command. ESPA begins analysis of the flight environment and takes proper action. If ESPA encounters an insufficient condition for each flight phase, the pilot receives a warning message. [Rep. 9:p. 92]

## ***2. Texas Instruments Emergency Procedures Expert System***

Texas Instruments concentrated on an Emergency Procedures Expert System, also for the F-16. This system deals with emergencies resulting from the loss of canopy and failure of the engine shaft providing power to the accessory gear boxes. The system uses the Explorer computer and LISP language. [Ref. 9:p. 92]

## ***3. McDonnell Douglas Avionics Integrated Maintenance Expert System***

Harris said "AIMES was developed to save time and money," and explained the need for a maintenance expert system:

As it stands now, when an avionics box goes off-line during a mission, a great deal of time has to be spent on the ground trying to isolate the failure so repairs can be made. Another maintenance problem is that in many instances the box will begin to work once the aircraft has landed. This results in the ground crew spending hours trying to duplicate the airborne failure so that the box can be repaired. [Ref. 10:p. 69]

AIMES operates by gathering F-18 aircraft data and creating flight files during missions for later analysis. Production rule-based artificial intelligence techniques are then used to isolate avionics failures to the electronic card level. The artificial intelligence expert system is initialized at the beginning of each flight. The system has an 80186 processor chip operating at 6 MHZ, 512 RAM, 256K EPROM, a 256K removable bubble cassette and a Mil-STD-1553 multiplexer controller. Also, AIMES has full internal built-in-test capability. Software features include forward and backward chaining capability and the inclusion of a resident rule debugger. The expert system is written in PLM. [Ref. 10:p. 69]

## **B. BACKGROUND OF THE TUTORING SYSTEM**

The training of military personnel is important to all the services. An important example of an intelligent computer-based training system in the military is STEAMER, developed by the Navy Personnel Research and Development Center to

teach propulsion engineering. An intelligent tutoring system like STEAMER shows that it is not necessary to have expensive equipment for students to learn the same material in a shorter period than with traditional instruction methods.

For our implementation, we'll use means-ends analysis, one of the classic techniques for solving search problems.

### *1. Means-ends Analysis*

Means-ends analysis is useful as a general problem solver in problems with many components [Ref. 11, 12]. It is capable of surprisingly powerful reasoning and is easy to customize for applications, using only a few lines in a Prolog program.

Means-ends analysis takes two inputs and returns two outputs. The first one is the starting state, the second is the goal state, the third is the list of operators found to solve the problem, and the fourth is the facts true in the final state. To reduce the difference between the starting state and the goal state, the means-ends analysis has to be a recursive search. The components of means-ends analysis are:

1. The recommended operator gives conditions for recommending an operator based on the facts different between the current state and the goal;
2. The precondition of recommended operator gives facts that must be true before applying an operator;
3. The deletepostcondition of recommended operator gives conditions that become no longer true after applying an operator;
4. The addpostcondition of recommended operator gives conditions that become true after applying an operator. [Ref. 13]

### *2. Means-ends tutor*

The idea of means-ends tutor is introduced in Rowe's technical report [Ref. 13].

Means-ends analysis is similar to the way people solve many action-sequence planning problems. Thus we can easily use means-ends analysis as a tutor. Specifically, our tutor kernel prompts the student for each action in turn



while using its own independent reasoning to select a best action for the current state. If the two actions disagree, the student is tutored as to the implications and disadvantages of the action. Six domain-independent bug types can be detected with student-selected actions. This checking can be interposed into the program just after the first recursive call of "means-ends" :

1. The student's action is a misspelling of a known action; point it out to the student and check for other errors with proposed correction.
2. The student's action is easily confusable with a recommended action at some point; point it out and check for other errors with the student's action.
3. The student's action does not satisfy preconditions of the action; tell the student what those preconditions are.
4. Student's action is an irreversible one: that means the problem cannot ever be solved if it is performed; warn student immediately, because it is a serious problem.
5. The student's action is useless; tell the student, but it may be more educational to wait until the student returns to a previous state or returns to the best path from the original state. The system can then point out the dimensions of the digression.
6. Both the student's action and the computer-selected action are possible in some state; the computer-selected action may be preferable by the priority order means-ends "recommended" facts. [Ref. 13]

### III. CONCEPT OF THE PILOT TUTORING SYSTEM

#### A. DESCRIPTION OF F-4 AIRCRAFT AND FUEL SYSTEM

##### *1. F-4 Aircraft*

The F-4 [Ref. 14:p. 1-1] aircraft design has been used for multi-purpose aircraft the since 1960s. There are two kinds of F-4 aircraft; the fighter and bomber such as F-4D, F-4E and the reconnaissance aircraft such as RF-4A, RF-4B and RF-4C. The F-4 aircraft is a two-place (tandem) supersonic, twin turbojet airplane built by McDonnell Douglas Corporation. The aircraft is designed for all-weather, high-low, day-night operations. It is powered by two single rotor, axial flow, variable stator turbo-jet J79-GE engines with afterburner. The aircraft feathers a low mounted swept-back wing with anhedral at the wing tips, and a one-piece stabilator with cathedral, mounted low on the aft fuselage.

##### *2. F-4 Fuel System*

The F-4 fuel system [Ref. 14: p. 1-107] consists of seven interconnected fuel cells in the fuselage, and two integral wet cells in the wing torque boxes. Provisions are made for two externally mounted droppable wing tanks and a droppable fuselage centerline external tank. Provisions are also made for an air refueling system. The function of fuselage cells 2, 3, 4, 5, 6, and 7 is to keep cell 1 supplied with fuel.

An air pressure fuel transfer system is provided to transfer wing and external tank fuel to the fuselage cells. Hydraulic and electric transfer pumps plus gravity feed are utilized to transfer fuel from the fuselage cells to the No.1 tank which is the engine feed tank.

All internal fuel cells incorporate capacitance-type fuel gaging units which continuously indicate the total fuel quantity in pounds in all internal cells. The fuel system is equipped with refueling level control valves which are float type valves that

shut off the pressure fueling when predetermined fuel levels are reached. All internal and external fuel tanks are pressurized in flight by regulated engine bleed air which is also utilized to transfer wing or external fuel to the fuselage cells or to dump wing fuel. And fuel is supplied to the engine during all flight attitudes by two submerged electric motor-driven centrifugal type boost pumps.

The fuel quantity indicator consists of a fuel counter indicator and a fuel sector indicator. The fuel counter indicator shows total quantity of internal wing fuel and fuselage fuel, and the fuel sector indicator shows quantity in the fuselage only. There is no indicator to represent the quantity in the external wing and centerline tanks, pilot must estimate those supplies.

## **B. PILOT PROCEDURE**

The pilot's procedure can usually be divided into normal and emergency procedures. A malfunction in any system will initiate the pilot's emergency procedure. To improve combat readiness and achieve a substantial reduction in the rate of aircraft accidents, emergency procedures were written into each aircraft's flight manual. The flight manual also includes performance data, flight characteristics and flight crew coordination. In the F-4 aircraft series, there is no major difference in fuel systems, so this thesis will use the RF-4B flight manual issued by Naval Training and Operating Procedures Standardization Program (NATOPS).

### ***1. Normal Procedure***

The normal procedure explains steps the pilot and flight crew must go through during each flight phase from briefing to debriefing. It covers the flight phase, aircraft configuration, and flight environment. The normal procedure goes through the sequence of preflight, takeoff, inflight, landing, postflight, scramble operation and night flying. Preflight procedures cover the time before entering the cockpit, after entering the cockpit, before starting engines, starting engines, before taxiing and taxiing. Takeoff procedures cover before takeoff, normal takeoff technique, minimum run

takeoff, crossword takeoff, formation takeoff, after takeoff, transition to climb, and climb. [Ref. 14:p. 3-1]

During normal procedures, flight conditions are good and there are few opportunities for decision making. Any mistakes by the pilot could cause problems necessitating emergency procedures.

## ***2. Emergency Procedure***

The emergency procedures ensure maximum safety for the crew and aircraft until a safe landing or other appropriate action is accomplished. In contrast to normal procedures, a pilot using emergency procedures must make decisions in a short time to fix the aircraft's problems.

During an emergency, the pilot must consider three things: the air crew must maintain aircraft control, the pilot must analyze the situation and take proper action and finally, the pilot must land as soon as practicable. The first and third procedure don't require any decision making or reasoning, but the second procedure can be difficult. [Ref. 14:p. 5-2]

The emergency procedures depend on the malfunctioning system, and what phase of the flight is involved. Emergency procedures cover air condition system malfunctions, electrical malfunction, engine malfunctions, fire, flight controls, fuel system, hydraulic system and landing emergencies.

## **C. OBJECTIVE**

The basic goal in an emergency is analyzing the situation exactly and taking proper action considering large amounts of data. The pilot emergency tutoring system teaches the pilot to:

1. Collect necessary information from lots of data.
2. Use necessary information and define the problem based on expert knowledge.



3. Identify the problem, and follow the emergency procedure for proper action.

#### **D. ASSUMPTIONS**

This thesis makes the following assumptions;

1. The status of a component in our tutoring system is the same as the actual aircraft status, even though there is no sensor for input data.
2. The flight data can't be changed without the pilot's action.
3. The current status of indicators will use input data not pilot readings.
4. The pilot is not allowed to handle any switch before identifying the problem, even if it is a useless switch.
5. If the pilot follows the correct procedure, the problem is cleared, even though there are mechanical problems that can't be fixed.

## **IV. IMPLEMENTATION OF THE PILOT TUTORING SYSTEM**

### **A. PROGRAMMING LANGUAGE**

To implement the pilot tutoring system, this study uses C-Prolog for an ISI workstation. The pilot tutoring system must compare current component states with goal states and reduce any differences to reach the goal state. This means-ends analysis requires the use of Prolog.

There are three kinds of Prolog language available at the Artificial Intelligence laboratory of Naval Postgraduate School; C-Prolog, M-Prolog and Quintus Prolog. C-Prolog and M-Prolog are used in the AI machine. Quintus Prolog is designed for a Sun operating system. Though the environment for each Prolog interpreter is different, there are few differences in usage from the user's point of view.

C-Prolog was designed for machines with a large uniform address space, and assumes a pointer cell 32 bits wide. This language can be used in VAX machines under the UNIX and VAX/VMS operating systems, on the Sun workstation under 4.1/2 UNIX, and the other MC68000-based workstations with minor changes. [Ref. 15]

### **B. ORGANIZATION OF THE PROGRAM**

The tutoring system model consists of a student model, a problem domain, an expert domain model, and a tutor domain model.

#### ***1. Student Model***

The student model for the tutoring must include a processing state for means-ends analysis. To track this, our tutor backtracks as necessary by exploiting built-in Prolog backtracking capability, to reach a processing state that explains why the student wants to do a particular action next. [Ref. 13:p. 4]

## ***2. Problem Model***

Aircraft emergencies can be caused by several reasons, but this study will focus on mechanical problems and pilot mistakes. The pilot can make two kinds of errors: a wrong decision or a wrong procedure. In the first error, the pilot doesn't correctly identify the problem, i.e., he confuses internal wing transfer failure with external wing transfer failure. Even if the pilot follows the correct procedure, it makes the problem harder to solve. The latter error means that the pilot follows the wrong procedure, even though he correctly diagnosed the problem.

A fuel system malfunction on an F-4 aircraft consists of fuel boost pump failure, fuel transfer failure and failed open defuel valve during air refueling. This study deals with transfer system failure. The fuel transfer failure could be internal wing fuel fails to transfer, centerline fuel fails to transfer, external wing fuel fails to transfer, reverse transfer of fuselage fuel, fuel streaming from underside of aircraft following center jettison and wing fuel leak. The tutoring system can't detect any other malfunction. It handles combined malfunctions on a priority basis.

## ***3. Expert Domain Model***

The expert domain of PETS can analyze a problem and take proper action within the F-4 fuel system malfunction. To analyze the problem, the pilot must look at the fuel quantity sector, fuel quantity counter, external tank, centerline tank and states of centerline jettison. After identifying the problem, the pilot will check the difference between the current state and goal state, and the system will suggest a corrective procedure based on the flight manual. Even though the final goal is the same in every emergency, each procedure is different. The expert domain consists of generic definitions to avoid program redundancy and basic components of the means-ends analysis.

#### ***4. Tutor Domain Model***

Each pilot action prompts a message such as "OK.," "It is not recommended right now," and "That operator has to satisfy these conditions." The tutoring domain model has two types: the first one teaches the mistakes of procedural action to the pilot, and the second one syntactical mistakes that may be corrected or not by tutoring system.

### **C. DATA STRUCTURES IN PROGRAM**

Usually, the data structure of means-ends analysis consists of facts <fact> and operators <operator>. A list of facts <factlist> represents conditions or the differences between current states and goal and a list of operators <oplist> represents collection of operators.

#### ***1. Input and Output***

To start the program, start states and goal states must be entered. The start states have to include five states of indicators and 20 states of switch status. Goal states always have to have an identified problem and a terminated problem. A state of components is created using random number selection to test student reliability. If a user wants to exercise a specific malfunction case, the start states can be given by the programmer to simulate that malfunction situation.

During the execution, the user must provide the best choice during each flight phase until the user reaches the goal state using his knowledge of the fuel system. When the user reaches the goal state, the tutoring system prints the entire user action sequence.

#### ***2. World Knowledge***

World knowledge consists of a definition of switch and indicator positions and operator definitions for means-ends analysis.



### *a. Definition of Switch and Indicator Position*

This study uses position and indication predicates to represent the values of the fuel system indicator and the position of the fuel system switch. This allows the use of means-ends analysis to determine a concrete solution based on a vague knowledge of the fuel system of F-4 aircraft. This is useful in:

1. Representing the knowledge of aircraft components as facts with ease.
2. Reducing redundant programs in the recommended, precondition, deletepostcondition and addpostcondition.
3. Finding out if certain names of switches or positions are available for our aircraft components.
4. Finding out the opposite position of the switch for precondition and deletepostcondition.

(1) Position. The position predicate consists of three or four arguments. The first argument represents the name of the switch, and second, third and fourth arguments represent the position of the switch. If there are three arguments in the switch, then it is a binary switch. If there are four arguments, then it is triple switch. The internal wing transfer switch is a binary switch which has normal and stop\_trans positions. An external wing transfer switch is a triple switch which has outbd, wing, and center positions.

**position(internal\_wing\_transfer, normal, stop\_trans).**  
**position(external\_wing\_transfer, outbd, wing, center).**

(2) Indication. The indication predicate consists of three or four arguments. The first argument represents the name of the indicator and second, third and fourth arguments represent the status of the indicator. The fuel quantity sector presents three status types: decrease, normal and increase.

**indication(fuel\_quantity\_sector, decrease, normal, increase).**



(3) Existposition. The existposition predicate reads two arguments. The first one is the switch name and the second one is the switch position. If that position exists in the switch, it will return true.

**existposition(internal\_wing\_transfer, normal).**

(4) Existindication. The existindication predicate reads two arguments; the first argument is the indicator name and the second argument is the indicator status.

**existindication(fuel\_quantity\_sector, decrease).**

*b. Means-ends Analysis*

This is quoted from Rowe's tutor program "METUTOR16."

(1) Recommended. The recommended predicate consists of two arguments, <difference list> and <operator>.

(2) Precondition. The precondition predicate consists of three or four arguments: <operator>, <conditional facts> and <precondition facts> for three arguments, and <operator>, <conditional facts>, <precondition facts> and <message> for four arguments.

(3) Deletepostcondition. The deletepostcondition predicate consists of three or four arguments: <operator>, <conditional facts>, <deleted facts> for three arguments, and <operator>, <conditional facts>, <deleted facts>, <message> for four arguments.

(4) Addpostcondition. The addpostcondition predicate consists of three or four arguments: <operator>, <conditional facts>, <added facts> for three arguments, and <operator>, <conditional facts>, <added facts> and <message> for four arguments.

### 3. Tutoring System

This is quoted from Rowe's technical report "Means-ends Tutoring." [Ref. 13]

#### *a. Tutor*

The tutor is top level of means-ends tutor, and consists of two arguments: start states <factlist> and goal <factlist>.

#### *b. Means\_ends\_tutor*

The means\_ends\_tutor consists of six arguments: current states <factlist>, goal <factlist>, operator list <oplist>, goal states <factlist>, stack and goal stack <factlist>.

#### *c. Check\_with\_student*

Check\_with\_student gets the operator from the student and consists of four arguments: student selected operator <operator>, current states <factlist>, the difference of current states and goal <factlist>, and computer-based choice <operator>.

#### *d. Handle\_student\_op*

Handle\_student\_op provides communication between students and the tutoring system. This consists of five arguments: student selected operator <operator>, computer-based operator <operator>, current states <factlist>, difference of current states and goal <factlist>, and student new selected operator <operator>.

#### *e. Writelist*

Writelist provides writing of lists. This consists of two arguments: list of facts or operators and the type of list.

#### *f. Once\_means\_ends*

This works a little differently from 'means\_ends\_tutor' in checking infinite loops for several situations. This consists of four arguments: start states <factlist>, goal <factlist>, operator list <oplist> and goalstate <factlist>.

## **D. IMPLEMENTATION OF THE EMERGENCY PROCEDURE**

### ***1. Simplification of the Global Variable***

Existposition reads the switch name and position. If the switch and position is available, it will return true, otherwise it will return false. Find\_opposite reads the switch name and position, then returns the opposite position of the switch. This is very important because in the binary switch if you want to move some switch position to another position, then the current position must be the precondition of another position and it will be the item of deletepostcondition. But, in the triple switch, we can't apply this find\_opposite predicate to find the opposite position. Instead, we use "not" predicate for the precondition and the other two positions for the items of deletepostcondition.

### ***2. Implementation of Means-ends Analysis in Emergency Procedure***

This part provides the translation of the expert's knowledge to the real application by the knowledge engineer. This part includes information on the pilot procedure and desired switch position in certain conditions. Throughout the program, the goals are the identified problem and terminate problem. In order to reach the goal, we have to choose the pilot action according to the emergency procedure in the flight manual.

#### ***a. Recommended Operator***

The recommended operator definition specifies the desired facts and the operator to reach the desired facts. There are two types of recommended operator: specific and generic. In specific operator, in order to get the identified problem status, we have to use an identify\_problem operator. In the generic recommended operator, in order to get the "X " position of switch "SW," we have to use "set" operator with argument "SW" and "X," if the switch and position are available in the F-4 aircraft.

**recommended([identified(problem)], identify\_problem).**

**recommended([status(SW,X)],set(SW,X)):-  
switch(SW),existposition(SW,X).**

***b. Precondition of Operator***

The precondition definition consists of an operator, conditional facts and a fact list of preconditions which have to be satisfied. There are two types of preconditions: procedural preconditions and generic preconditions. The procedural precondition has several facts that the operator must apply under the identified problem, though the generic precondition has only one, the opposite position of that switch.

**precondition(identify\_problem,\_,  
[looked\_at(fuel\_quantity\_sector),  
looked\_at(fuel\_quantity\_counter),  
looked\_at(external\_tank),  
looked\_at(centerline\_tank),  
looked\_at(center\_tank\_jettison)]).**

**precondition(set(internal\_wing\_transfer, normal),  
[problem(internal\_wing\_transfer\_failure)],  
[identified(problem),  
status(internal\_wing\_transfer, stop\_trans),  
status(external\_transfer, off)]).**

**precondition(set(SW,X),\_,[status(SW,OX), identified(problem)]):-  
binary\_switch(SW), find\_opposite(SW,O,OX).**

**precondition(set(SW,X),\_,[identified(problem), not status(SW,X)]):-  
tri\_switch(SW).**

We have to satisfy these conditions to apply identify\_problem, looked\_at fuel\_quantity\_sector, looked\_at fuel\_quantity\_counter, looked\_at external\_tank and looked\_at center\_tank\_jettison. Usually, a procedural precondition has two components: an actual procedural precondition and existence precondition. The preconditions of "set(internal\_wing\_transfer, normal)" are identified problem,



"status(external\_transfer, stop\_trans)" which is a procedural precondition, and "status(internal\_wing\_transfer, stop\_trans)" which is the opposite position of the current position and is automatically an existence precondition.

If there is no declared problem, the second argument of the condition clause will be filled with an empty list. The precondition of the operator needs the opposite position of the current position in a binary switch and any position except the current position in a triple switch.

Preconditions are important, because they decide the sequence of a pilot procedure. Usually there is only one appropriate action for the pilot emergency procedure. But means-ends analysis can explore several ways to solve the problem. This can be multiprocessing, but it is more confusing to the pilot, so we used a simpler way.

#### *c. Deletepostcondition of Operator*

The deletepostcondition definition consists of an operator, condition facts and delete items. The find\_opposite predicate is useful with a binary switch. But the external\_wing\_transfer switch has a triple position; if we want to set the switch to "wing," then we have to delete either a "center" or "outbd" position.

```
deletepostcondition(set(external_wing_transfer,wing),_,
    [status(external_wing_transfer, center),
     status(external_wing_transfer, outbd)]).
```

```
deletepostcondition(set(SW,X),_,[status(SW,OX)]):-
    binary_switch(SW), find_opposite(SW,X,OX).
```

#### *d. Addpostcondition of Operator*

The addpostcondition definition consists of operator, condition facts and add items. It is simpler than precondition and deletepostcondition because there is no need for an opposite position. The "identify\_problem" reads four or five indicators from the look\_at operator, then adds some problem facts. For the example

below, if the fuel quantity sector indicates decrease, the fuel quantity counter indicates decrease, the external tank is empty, and the centerline tank is empty, then addpostcondition adds an internal-wing-transfer-failure problem fact.

```
addpostcondition(set(SW,X),_,[status(SW,X)]):-  
    switch(SW),existpostion(SW,X).
```

```
addpostcondition(identify_problem,  
    [indicator(fuel_quantity_sector,decrease),  
    indicator(fuel_quantity_counter,decrease),  
    indicator(external_tank,empty),  
    indicator(centerline_tank,empty)],  
    [problem(internal_wing_transfer_failure),  
    identified(problem)]).
```

## E. IMPLEMENTATION OF THE TUTOR

This tutoring program provides general tutoring for a means-ends analysis approach. Means-ends analysis is similar to the way people solve many action-sequence planning problems. Thus, a simple way to turn means-ends analysis into a tutor is to run it in approximate synchrony with student. Our tutor kernel prompts the student for each action in turn while using its own independent reasoning to select a best action for the current state. This was originally written by Rowe [Ref. 13], but was modified to implement our program, specifically for handling output.

### *1. Student Execution Error*

During the execution, `check_with_student` implements the tutoring strategies for different kinds of student errors. Some are straightforward such as spelling errors and precondition violations, but others require complex analysis with calls to 'means\_ends' on hypothetical states created by the student.

## *2. Natural Language*

The predicate writelist is used for the natural language. The writelist takes two arguments: fact list or operator list and type of list, since output is printed differently depending on whether it is states, preconditioned lists, or operators. In writelist2, we initialize the item number and total character numbers of previous item. If the item number is odd, it is noted at the beginning of the line. If the item number is even, it is noted in column 40 of the line.

## V. ANALYSIS

As previously mentioned, the pilot emergency tutoring system provides test cases for six major emergencies in the fuel system. To test it, we tested the six emergency cases, the normal case, a general case using total switches in the fuel system, and a random selecting case with computer-generated random numbers.

Table I presents the global and local stack and CPU runtime of the program. Sometimes it is necessary to extend the global stack space to run the program. Debugflag in the reverse failure and general cases could not be used without the extension.

The CPU run time ranges from 48.77 sec for test streaming from the underside of aircraft following center jettison to 188.20 sec for the general test case on an ISI workstation. Since the reverse transfer of fuselage fuel is the most complicated case and has embedded procedures, the CPU run time is 99.52 sec. The general case test similar to the reverse transfer of fuselage fuel must compare all switch positions with goal state, not the related switches. There is no difference in pilot procedure, but the CPU run time is increased to 188.20 sec. But a general case must be used in applying means-ends analysis in real life.

Two machines were used to test the tutoring system: an ISI workstation and a SPARC station. In the test of internal wing fuel failing to transfer, the ISI workstation took 52.23 sec, while the SPARC station took 15.23 sec. The SPARC station is faster than the ISI workstation in every test case.

During the test, students made many mistakes in writing the names of component and status. Some cases can be accepted, while in other cases, the student has to type the correct name of the component and status again. During the test, if students have a different solution than the computer-based solution, then the student answer will be regarded wrong.



Porter's [Ref. 7] previous work is comparable. It used means-ends analysis to handle aircraft emergencies, but did not tutor. CPU run time differs: Porter's expert system took 0.39 sec to 2.53 using the Turbo Prolog in PC. Our tutoring system takes 14.25 sec to 53.92 sec using C-Prolog in SPARC station. It is not fair to compare CPU run times, because our tutoring system is a complicated and embedded program. Another major difference is that our system uses two high-level objectives of identified problem and terminated problem.

TABLE I RESULT OF OUTPUT

Emergency \ Items		CPU Run Time	Global Stack	Local Stack
Internal Wing Fuel Failure	ISI	52.23 Sec	280 K	23.5 K
	SPARC	15.23 Sec	281 K	23.5 K
Centerline Fuel Failure	ISI	64.05 Sec	320 K	25.8 K
	SPARC	18.73 Sec	320 K	25.8 K
External Wing Fuel Failure	ISI	67.58 Sec	313 K	25.9 K
	SPARC	19.62 Sec	313 K	25.9 K
Reverse Transfer of Fuselage	ISI	99.52 Sec	479 K	31.9 K
	SPARC	28.62 Sec	479 K	31.9 K
Streaming from Underside	ISI	48.77 Sec	249 K	22.7 K
	SPARC	14.25 Sec	249 K	22.7 K
Wing Fuel Leaking	ISI	51.15 Sec	221 K	20.2 K
	SPARC	13.40 Sec	217 K	20.2 K
Normal situation	ISI	53.28 Sec	378K	23.3 K
	SPARC	15.45 Sec	378 K	23.3 K
General case	ISI	188.20 Sec	1042 K	42.3 K
	SPARC	53.92 Sec	1042 K	42.3 K
Random Selecting Case	ISI	47.70 Sec	328 K	29.4 K
	SPARC	13.77 Sec	330 K	29.2 K

## **VI. CONCLUSIONS**

### **A. CONCLUSIONS**

The pilot emergency tutoring system uses an intelligent education program to reduce pilot errors during an inflight emergency. It uses knowledge of the fuel system to diagnose and correct problems. By helping the pilot analyze the problem and determine how to fix it, the pilot can learn to avoid wrong decisions and procedures. By adding other airplane systems to the program, such as the engine, and electrical and hydraulic systems, the pilot overload from decision-making could be reduced. Consequently, the pilot can be free to serve as aircraft commander and can reduce the decision and procedure errors.

This pilot tutoring system can run on any system which can use C-Prolog. Using natural language makes it a user-friendly system. The debug flag feature provides a development program for the programmer.

### **B. FURTHER WORK**

Further work on a computer tutoring system could involve:

1. Combining other component emergencies such as electrical failure, hydraulic failure and engine failure in order to make a whole aircraft tutoring system.
2. Combining the knowledge of diagnostics in an aircraft maintenance expert system into the pilot emergency tutoring system.
3. Using the graphics to read input from students and write output, more like a simulator, to improve the efficiency of education.

## APPENDIX A

### TEST OF EMERGENCY CASE

#### A. INTERNAL WING FUEL FAILS TO TRANSFER

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.86667 sec.

tutor consulted 38916 bytes 4.35 sec.

yes

| ?- internal.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

external transfer is off, internal wing transfer is stop trans,  
refuel probe is extended, cb internal wing transfer is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at, external transfer is off,  
internal wing transfer is stop trans, refuel probe is extended,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.



What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at,      centerline tank is looked at,  
external transfer is off,              internal wing transfer is stop trans,  
refuel probe is extended,              cb internal wing transfer is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,              external transfer is off,  
internal wing transfer is stop trans,      refuel probe is extended,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,              centerline tank is looked at,  
external transfer is off,              internal wing transfer is stop trans,  
refuel probe is extended,              cb internal wing transfer is pulled,

and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external transfer is off,  
internal wing transfer is stop trans, refuel probe is extended,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: identify\_problem.

OK.

Failure of internal wing fuel to transfer caused by:

1. The wing tanks failing to pressurized
2. The wing transfer valves failing to open

\*\*\*\*\*

The following facts are now true:

internal wing transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external transfer is off,  
internal wing transfer is stop trans, refuel probe is extended,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set internal\_wing\_transfer normal.

OK.

\*\*\*\*\*

The following facts are now true:

internal wing transfer is normal,      internal wing transfer fail is problem,  
problem is identified,              external tank is looked at,  
fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
external transfer is off,              refuel probe is extended,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH, POSITION)

|: set refuel\_probe retracted.

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is retracted,              internal wing transfer is normal,  
internal wing transfer fail is problem, problem is identified,  
external tank is looked at,              fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,              external transfer is off,  
cb internal wing transfer is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH, POSITION)

|: set cb\_internal\_wing\_transfer pushed.

OK.

\*\*\*\*\*

The following facts are now true:

cb internal wing transfer is pushed,      refuel probe is retracted,  
internal wing transfer is normal,      internal wing transfer fail is problem,  
problem is identified,              external tank is looked at,  
fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,

external transfer is off,            and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set wing\_transfer\_pressure ovrdr\_trans.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovrdr\_trans,    cb internal wing transfer is pushed,

refuel probe is retracted,            internal wing transfer is normal,

internal wing transfer fail is problem, problem is identified,

external tank is looked at,            fuel quantity counter is looked at,

fuel quantity sector is looked at,    centertank jettison is looked at,

centerline tank is looked at,        and external transfer is off.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,            look\_at centertank\_jettison,

look\_at fuel\_quantity\_sector,       look\_at fuel\_quantity\_counter,

look\_at external\_tank,            identify\_problem,

internal\_wing\_transfer to normal,    refuel\_probe to retracted,

cb\_internal\_wing\_transfer to pushed,   wing\_transfer\_pressure to ovrdr\_trans,

and terminate\_problem

yes

! ?- statistics.

atom space: 96K (in use: 36064, max. used: 36064)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2624)

heap: 1000K (in use: 122540, max. used: 189732)

global stack: 1900K (in use: 0, max. used: 280308)

local stack: 1000K (in use: 300, max. used: 23568)



Runtime: 52.23 sec.

yes

! ?- halt.

[ Prolog execution halted ]

## B. CENTERLINE FUEL FAILS TO TRANSFER

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.83333 sec.

tutor consulted 38916 bytes 4.23333 sec.

yes

| ?- center.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

external wing transfer is wing,      cb external wing control is pulled,  
cb left fuel valve power is pulled,      refuel probe is extended,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at,      external wing transfer is wing,  
cb external wing control is pulled,      cb left fuel valve power is pulled,  
refuel probe is extended,      and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at,      centerline tank is looked at,  
external wing transfer is wing,      cb external wing control is pulled,  
cb left fuel valve power is pulled,      refuel probe is extended,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      external wing transfer is wing,  
cb external wing control is pulled,      cb left fuel valve power is pulled,  
refuel probe is extended,      and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
external wing transfer is wing,      cb external wing control is pulled,  
cb left fuel valve power is pulled,      refuel probe is extended,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external wing transfer is wing,  
cb external wing control is pulled, cb left fuel valve power is pulled,  
refuel probe is extended, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: identify\_problem.

OK.

Failure of the centerline fuel to transfer caused by:

1. The defueling shutoff valve failing to open position
2. The refueling valve failing to the closed position
3. The tank failing to become pressurized

\*\*\*\*\*

The following facts are now true:

centerline transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external wing transfer is wing,  
cb external wing control is pulled, cb left fuel valve power is pulled,  
refuel probe is extended, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set\_external\_wing\_transfer\_center.

OK.

\*\*\*\*\*

The following facts are now true:

external wing transfer is center, centerline transfer fail is problem,  
problem is identified, external tank is looked at,



fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
cb external wing control is pulled, cb left fuel valve power is pulled,  
refuel probe is extended, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_external\_wing\_control pushed.

OK.

\*\*\*\*\*

The following facts are now true:

cb external wing control is pushed, external wing transfer is center,  
centerline transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, cb left fuel valve power is pulled,  
refuel probe is extended, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_left\_fuel\_valve\_power pushed.

OK.

\*\*\*\*\*

The following facts are now true:

cb left fuel valve power is pushed, cb external wing control is pushed,  
external wing transfer is center, centerline transfer fail is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
refuel probe is extended, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set refuel\_probe retracted.

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is retracted,                cb left fuel valve power is pushed,

cb external wing control is pushed,    external wing transfer is center,

centerline transfer fail is problem,    problem is identified,

external tank is looked at,            fuel quantity counter is looked at,

fuel quantity sector is looked at,    centertank jettison is looked at,

centerline tank is looked at,    and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set wing\_transfer\_pressure ovr\_d\_trans.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovr\_d\_trans,    refuel probe is retracted,

cb left fuel valve power is pushed,    cb external wing control is pushed,

external wing transfer is center,    centerline transfer fail is problem,

problem is identified,                external tank is looked at,

fuel quantity counter is looked at,    fuel quantity sector is looked at,

centertank jettison is looked at,    and centerline tank is looked at.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,                look\_at centertank\_jettison,

look\_at fuel\_quantity\_sector,           look\_at fuel\_quantity\_counter,

look\_at external\_tank,                identify\_problem,

external\_wing\_transfer to center,    cb\_external\_wing\_control to pushed,

cb\_left\_fuel\_valve\_power to pushed, refuel\_probe to retracted,  
wing\_transfer\_pressure to ovrd\_trans,and terminate\_problem  
yes

! ?- statistics.

atom space: 96K (in use: 36016, max. used: 36016)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2772)

heap: 1000K (in use: 128292, max. used: 204184)

global stack: 1900K (in use: 0, max. used: 320508)

local stack: 1000K (in use: 300, max. used: 25896)

Runtime: 64.05 sec.

yes

! ?- halt.

[ Prolog execution halted ]

### C. EXTERNAL WING FUEL FAILS TO TRANSFER

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.85 sec.

tutor consulted 38916 bytes 4.21667 sec.

yes

| ?- external.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

external wing transfer is center,      cb external wing control is pulled,

refuel probe is extended,      cb fuel valve power is pulled,

and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at,      external wing transfer is center,

cb external wing control is pulled,      refuel probe is extended,

cb fuel valve power is pulled,      and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centertank\_jettison.

OK.



\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at,      centerline tank is looked at,  
external wing transfer is center,      cb external wing control is pulled,  
refuel probe is extended,      cb fuel valve power is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      external wing transfer is center,  
cb external wing control is pulled,      refuel probe is extended,  
cb fuel valve power is pulled,      and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
external wing transfer is center,      cb external wing control is pulled,  
refuel probe is extended,      cb fuel valve power is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external wing transfer is center,  
cb external wing control is pulled, refuel probe is extended,  
cb fuel valve power is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: identify\_problem.

OK.

Failure of the external wing fuel to transfer caused by:

1. The external wing shutoff valve failing to the closed
2. The tanks failing to become pressurized

\*\*\*\*\*

The following facts are now true:

external wing transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, external wing transfer is center,  
cb external wing control is pulled, refuel probe is extended,  
cb fuel valve power is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set external\_wing\_transfer outbd.

OK.

\*\*\*\*\*

The following facts are now true:

external wing transfer is outbd, external wing transfer fail is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,

centertank jettison is looked at, centerline tank is looked at,  
cb external wing control is pulled, refuel probe is extended,  
cb fuel valve power is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_external\_wing\_control pushed.

OK.

\*\*\*\*\*

The following facts are now true:

cb external wing control is pushed, external wing transfer is outbd,

external wing transfer fail is problem, problem is identified,

external tank is looked at, fuel quantity counter is looked at,

fuel quantity sector is looked at, centertank jettison is looked at,

centerline tank is looked at, refuel probe is extended,

cb fuel valve power is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set refuel\_probe retracted.

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is retracted, cb external wing control is pushed,

external wing transfer is outbd, external wing transfer fail is problem,

problem is identified, external tank is looked at,

fuel quantity counter is looked at, fuel quantity sector is looked at,

centertank jettison is looked at, centerline tank is looked at,

cb fuel valve power is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_fuel\_valve\_power pushed.

OK.

\*\*\*\*\*

The following facts are now true:

cb fuel valve power is pushed,      refuel probe is retracted,  
cb external wing control is pushed,      external wing transfer is outbd,  
external wing transfer fail is problem, problem is identified,  
external tank is looked at,      fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set wing\_transfer\_pressure ovr\_d\_trans.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovr\_d\_trans,      cb fuel valve power is pushed,  
refuel probe is retracted,      cb external wing control is pushed,  
external wing transfer is outbd,      external wing transfer fail is problem,  
problem is identified,      external tank is looked at,  
fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      and centerline tank is looked at.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,      look\_at centertank\_jettison,  
look\_at fuel\_quantity\_sector,      look\_at fuel\_quantity\_counter,  
look\_at external\_tank,      identify\_problem,  
external\_wing\_transfer to outbd,      cb\_external\_wing\_control to pushed,  
refuel\_probe to retracted,      cb\_fuel\_valve\_power to pushed,  
wing\_transfer\_pressure to ovr\_d\_trans,and terminate\_problem



yes

! ?- statistics.

atom space: 96K (in use: 36016, max. used: 36016)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2744)

heap: 1000K (in use: 129820, max. used: 203072)

global stack: 1900K (in use: 0, max. used: 313440)

local stack: 1000K (in use: 300, max. used: 25996)

Runtime: 67.58 sec.

yes

! ?- halt.

[ Prolog execution halted ]

## D. REVERSE TRANSFER OF FUSELAGE FUEL

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.81667 sec.

tutor consulted 38916 bytes 4.26667 sec.

yes

| ?- reverse.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

ram air turbine is retracted, generators is on,

cb wing transfer control is pushed, cb left boost pump control is pushed,

cb boost pump emer control is pushed, cb right boost pump control is pulled,

and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at, ram air turbine is retracted,

generators is on, cb wing transfer control is pushed,

cb left boost pump control is pushed, cb boost pump emer control is pushed,

cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
ram air turbine is retracted, generators is on,  
cb wing transfer control is pushed, cb left boost pump control is pushed,  
cb boost pump emer control is pushed, cb right boost pump control is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, ram air turbine is retracted,  
generators is on, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
ram air turbine is retracted, generators is on,  
cb wing transfer control is pushed, cb left boost pump control is pushed,  
cb boost pump emer control is pushed, cb right boost pump control is pulled,  
and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, ram air turbine is retracted,  
generators is on, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : identify\_problem.

OK.

Reverse transfer of fuselage fuel caused by:

A failed open defuel valve

\*\*\*\*\*

The following facts are now true:

reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, ram air turbine is retracted,  
generators is on, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set ram\_air\_turbine extended.



OK.

\*\*\*\*\*

The following facts are now true:

ram air turbine is extended,            reverse transfer fail is problem,  
problem is identified,            external tank is looked at,  
fuel quantity counter is looked at,    fuel quantity sector is looked at,  
centertank jettison is looked at,    centerline tank is looked at,  
generators is on,            cb wing transfer control is pushed,  
cb left boost pump control is pushed,    cb boost pump emer control is pushed,  
cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: set generators off.

OK.

\*\*\*\*\*

The following facts are now true:

generators is off,            ram air turbine is extended,  
reverse transfer fail is problem,    problem is identified,  
external tank is looked at,    fuel quantity counter is looked at,  
fuel quantity sector is looked at,    centertank jettison is looked at,  
centerline tank is looked at,    cb wing transfer control is pushed,  
cb left boost pump control is pushed,    cb boost pump emer control is pushed,  
cb right boost pump control is pulled, and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_left\_boost\_pump\_control pulled.

OK.

\*\*\*\*\*

The following facts are now true:

cb left boost pump control is pulled,    generators is off,  
ram air turbine is extended,    reverse transfer fail is problem,  
problem is identified,    external tank is looked at,

fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
cb wing transfer control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled,and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set cb\_boost\_pump\_emer\_control pulled.

OK.

\*\*\*\*\*

The following facts are now true:

cb boost pump emer control is pulled, cb left boost pump control is pulled,  
generators is off, ram air turbine is extended,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, cb wing transfer control is pushed,  
cb right boost pump control is pulled,and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set generators on.

OK.

\*\*\*\*\*

The following facts are now true:

generators is on, cb boost pump emer control is pulled,  
cb left boost pump control is pulled, ram air turbine is extended,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, cb wing transfer control is pushed,  
cb right boost pump control is pulled,and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: set ram\_air\_turbine retracted.

OK.

\*\*\*\*\*

The following facts are now true:

ram air turbine is retracted, generators is on,  
cb boost pump emer control is pulled, cb left boost pump control is pulled,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, cb wing transfer control is pushed,  
cb right boost pump control is pulled,and wing transfer pressure is normal.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: set wing\_transfer\_pressure ovrd\_trans.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovrd trans, ram air turbine is retracted,  
generators is on, cb boost pump emer control is pulled,  
cb left boost pump control is pulled, reverse transfer fail is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
cb wing transfer control is pushed,and cb right boost pump control is pulled.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank, look\_at centertank\_jettison,

look\_at fuel\_quantity\_sector,        look\_at fuel\_quantity\_counter,  
look\_at external\_tank,        identify\_problem,  
ram\_air\_turbine to extended,        generators to off,  
cb\_left\_boost\_pump\_control to pulled,    cb\_boost\_pump\_emer\_control to pulled,  
generators to on,        ram\_air\_turbine to retracted,  
wing\_transfer\_pressure to ovrdr\_trans,and terminate\_problem  
yes

! ?- statistics.

atom space: 96K (in use: 36092, max. used: 36092)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2960)

heap: 1000K (in use: 149036, max. used: 231496)

global stack: 1900K (in use: 0, max. used: 479100)

local stack: 1000K (in use: 300, max. used: 31900)

Runtime: 99.52 sec.

yes

! ?- halt.

[ Prolog execution halted ]



## E. FUEL STREAMING FROM UNDERSIDE OF AIRCRAFT FOLLOWING CENTER JETTISON

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.86667 sec.

tutor consulted 38916 bytes 4.26667 sec.

yes

| ?- stream.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovrd trans,    cb refuel probe is pushed,  
refuel probe is extended,                and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at,            wing transfer pressure is ovrd trans,  
cb refuel probe is pushed,                refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
wing transfer pressure is ovrdrans, cb refuel probe is pushed,  
refuel probe is extended, and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, wing transfer pressure is ovrdrans,  
cb refuel probe is pushed, refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
wing transfer pressure is ovrdrans, cb refuel probe is pushed,  
refuel probe is extended, and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, wing transfer pressure is ovrd trans,  
cb refuel probe is pushed, refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : identify\_problem.

OK.

Fuel streaming from underside of aircraft following CL jett

1. Cockpit indications are identical to reverse transfer
2. Fuel loss on the sector and counter: 500~700 lbs/min

\*\*\*\*\*

The following facts are now true:

fuel streaming underside is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, wing transfer pressure is ovrd trans,  
cb refuel probe is pushed, refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set wing\_transfer\_pressure normal.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is normal, fuel streaming underside is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,

cb refuel probe is pushed,            refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

**|: set cb\_refuel\_probe pulled.**

OK.

\*\*\*\*\*

The following facts are now true:

cb refuel probe is pulled,            wing transfer pressure is normal,  
fuel streaming underside is problem,    problem is identified,  
external tank is looked at,           fuel quantity counter is looked at,  
fuel quantity sector is looked at,    centertank jettison is looked at,  
centerline tank is looked at,        refuel probe is extended,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

**|: set refuel\_probe refuel.**

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is refuel,            cb refuel probe is pulled,  
wing transfer pressure is normal,    fuel streaming underside is problem,  
problem is identified,            external tank is looked at,  
fuel quantity counter is looked at,    fuel quantity sector is looked at,  
centertank jettison is looked at,    centerline tank is looked at,  
and cb fuel boost pump is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

**|: set cb\_fuel\_boost\_pump pulled.**

OK.



\*\*\*\*\*

The following facts are now true:

cb fuel boost pump is pulled,            refuel probe is refuel,  
cb refuel probe is pulled,            wing transfer pressure is normal,  
fuel streaming underside is problem,    problem is identified,  
external tank is looked at,           fuel quantity counter is looked at,  
fuel quantity sector is looked at,    centertank jettison is looked at,  
and centerline tank is looked at.

What operator do you choose?

For example

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,            look\_at centertank\_jettison,  
look\_at fuel\_quantity\_sector,       look\_at fuel\_quantity\_counter,  
look\_at external\_tank,            identify\_problem,  
wing\_transfer\_pressure to normal,    cb\_refuel\_probe to pulled,  
refuel\_probe to refuel,            cb\_fuel\_boost\_pump to pulled,  
and terminate\_problem

yes

! ?- statistics.

atom space: 96K (in use: 35940, max. used: 35940)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2468)

heap: 1000K (in use: 119428, max. used: 177800)

global stack: 1900K (in use: 0, max. used: 249788)

local stack: 1000K (in use: 300, max. used: 22780)

Runtime: 48.77 sec.

yes

! ?- halt.

[ Prolog execution halted ]

## F. WING FUEL LEAK

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.85 sec.

tutor consulted 38916 bytes 4.25 sec.

yes

| ?- leaking.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

l: look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

l: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_counter.

Not the operator I would choose, but let us try it.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,    centertank jettison is looked at,  
centerline tank is looked at,    and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,    fuel quantity counter is looked at,  
centertank jettison is looked at,    centerline tank is looked at,  
and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at,    fuel quantity sector is looked at,  
fuel quantity counter is looked at,    centertank jettison is looked at,  
centerline tank is looked at,    and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: identify\_problem.

OK.

Wing fuel leak problem caused by:

Mechanical problem

\*\*\*\*\*

The following facts are now true:

wing fuel leaking is problem,            problem is identified,  
external tank is looked at,            fuel quantity sector is looked at,  
fuel quantity counter is looked at,    centertank jettison is looked at,  
centerline tank is looked at,    and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

|: disengage\_afterburner.

OK.

\*\*\*\*\*

The following facts are now true:

disengage is afterburner,            wing fuel leaking is problem,  
problem is identified,            external tank is looked at,  
fuel quantity sector is looked at,    fuel quantity counter is looked at,  
centertank jettison is looked at,    centerline tank is looked at,  
and refuel probe is retracted.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

!: fly\_below\_threehundredsfts.

OK.

\*\*\*\*\*

The following facts are now true:

below threehundredsfts is flight,    disengage is afterburner,  
wing fuel leaking is problem,            problem is identified,  
external tank is looked at,            fuel quantity sector is looked at,  
fuel quantity counter is looked at,    centertank jettison is looked at,  
centerline tank is looked at,    and refuel probe is retracted.



What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set refuel\_probe refuel.

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is refuel, below threehundredsfts is flight,

disengage is afterburner, wing fuel leaking is problem,

problem is identified, external tank is looked at,

fuel quantity sector is looked at, fuel quantity counter is looked at,

centertank jettison is looked at, and centerline tank is looked at.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: land as \_soon\_ as \_practicable.

OK.

\*\*\*\*\*

The following facts are now true:

as soon as practicable is landing, refuel probe is refuel,

below threehundredsfts is flight, disengage is afterburner,

wing fuel leaking is problem, problem is identified,

external tank is looked at, fuel quantity sector is looked at,

fuel quantity counter is looked at, centertank jettison is looked at,

and centerline tank is looked at.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: land arrested.

OK.

\*\*\*\*\*

The following facts are now true:

arrested is landing, as soon as practicable is landing,

refuel probe is refuel,                      below threehundredsks is flight,  
disengage is afterburner,                      wing fuel leaking is problem,  
problem is identified,                      external tank is looked at,  
fuel quantity sector is looked at,                      fuel quantity counter is looked at,  
centertank jettison is looked at,                      and centerline tank is looked at.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

**! : terminate\_problem.**

OK.

Full list of your actions:

look\_at centerline\_tank,                      look\_at centertank\_jettison,  
look\_at fuel\_quantity\_counter,                      look\_at fuel\_quantity\_sector,  
look\_at external\_tank,                      identify\_problem,  
disengage afterburner,                      fly below\_threehundredsks,  
refuel\_probe to refuel,                      land as\_soon\_as\_practicable,  
land arrested,                      and terminate\_problem

yes

| ?- statistics.

atom space: 96K (in use: 35828, max. used: 35828)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 2500)

heap: 1000K (in use: 121028, max. used: 175880)

global stack: 1900K (in use: 0, max. used: 221260)

local stack: 1000K (in use: 300, max. used: 20284)

Runtime: 51.15 sec.

yes

| ?- halt.

[ Prolog execution halted ]

## G. NORMAL SITUATION

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.81667 sec.

tutor consulted 38916 bytes 4.21667 sec.

yes

| ?- normal.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

internal wing transfer is normal,      external transfer is off,  
external wing transfer is outbd,      wing transfer pressure is normal,  
wing dump is normal,      refuel selection is int only,  
landing gear is up,      ram air turbine is retracted,  
boost pump left is normal,      boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
generators is on,      refuel probe is retracted,  
cb external wing control is pushed,      cb internal wing transfer is pushed,  
cb fuel valve power is pushed,      cb left fuel valve power is pushed,  
cb wing transfer control is pushed,      cb left boost pump control is pushed,  
cb boost pump emer control is pushed,      cb right boost pump control is pulled,  
cb fuel boost pump is pushed,      and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at, internal wing transfer is normal,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is normal, wing dump is normal,  
refuel selection is int only, landing gear is up,  
ram air turbine is retracted, boost pump left is normal,  
boost pump right is normal, external store jettison center is norm,  
external store jettison wing is norm, generators is on,  
refuel probe is retracted, cb external wing control is pushed,  
cb internal wing transfer is pushed, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

!: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
internal wing transfer is normal, external transfer is off,  
external wing transfer is outbd, wing transfer pressure is normal,  
wing dump is normal, refuel selection is int only,  
landing gear is up, ram air turbine is retracted,  
boost pump left is normal, boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
generators is on, refuel probe is retracted,  
cb external wing control is pushed, cb internal wing transfer is pushed,  
cb fuel valve power is pushed, cb left fuel valve power is pushed,  
cb wing transfer control is pushed, cb left boost pump control is pushed,  
cb boost pump emer control is pushed, cb right boost pump control is pulled,  
cb fuel boost pump is pushed, and cb refuel probe is pushed.

What operator do you choose?

For example:



type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)  
|: look\_at fuel\_quantity\_sector.  
OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      internal wing transfer is normal,  
external transfer is off,      external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,      landing gear is up,  
ram air turbine is retracted,      boost pump left is normal,  
boost pump right is normal,      external store jettison center is norm,  
external store jettison wing is norm,      generators is on,  
refuel probe is retracted,      cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
cb right boost pump control is pulled,      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)  
|: look\_at fuel\_quantity\_counter.  
OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
internal wing transfer is normal,      external transfer is off,  
external wing transfer is outbd,      wing transfer pressure is normal,  
wing dump is normal,      refuel selection is int only,  
landing gear is up,      ram air turbine is retracted,  
boost pump left is normal,      boost pump right is normal,  
external store jettison center is norm,      external store jettison wing is norm,

generators is on,                      refuel probe is retracted,  
cb external wing control is pushed,      cb internal wing transfer is pushed,  
cb fuel valve power is pushed,              cb left fuel valve power is pushed,  
cb wing transfer control is pushed,      cb left boost pump control is pushed,  
cb boost pump emer control is pushed,      cb right boost pump control is pulled,  
cb fuel boost pump is pushed,      and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at,              fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,              internal wing transfer is normal,  
external transfer is off,              external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,              landing gear is up,  
ram air turbine is retracted,              boost pump left is normal,  
boost pump right is normal,              external store jettison center is norm,  
external store jettison wing is norm,      generators is on,  
refuel probe is retracted,              cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
cb right boost pump control is pulled,      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : identify\_problem.

OK.

There is no problem.

Full list of your actions:

look\_at centerline\_tank, look\_at centertank\_jettison,  
look\_at fuel\_quantity\_sector, look\_at fuel\_quantity\_counter,  
look\_at external\_tank, and identify\_problem

yes

! ?- statistics.

atom space: 96K (in use: 37040, max. used: 37040)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 1916)

heap: 1000K (in use: 117756, max. used: 148816)

global stack: 1900K (in use: 0, max. used: 378224)

local stack: 1000K (in use: 300, max. used: 23344)

Runtime: 53.28 sec.

yes

! ?- halt.

## H. GENERAL CASE

C-Prolog version 1.5

| ?- [expert,tutor].

expert consulted 19672 bytes 2.83333 sec.

tutor consulted 38916 bytes 4.26667 sec.

yes

| ?- general.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

internal wing transfer is normal,      external transfer is off,  
external wing transfer is outbd,      wing transfer pressure is normal,  
wing dump is normal,      refuel selection is int only,  
landing gear is up,      ram air turbine is retracted,  
boost pump left is normal,      boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
generators is on,      refuel probe is retracted,  
cb external wing control is pushed,      cb internal wing transfer is pushed,  
cb fuel valve power is pushed,      cb left fuel valve power is pushed,  
cb wing transfer control is pushed,      cb left boost pump control is pushed,  
cb boost pump emer control is pushed,      cb right boost pump control is pulled,  
cb fuel boost pump is pushed,      and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:



centerline tank is looked at, internal wing transfer is normal,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is normal, wing dump is normal,  
refuel selection is int only, landing gear is up,  
ram air turbine is retracted, boost pump left is normal,  
boost pump right is normal, external store jettison center is norm,  
external store jettison wing is norm, generators is on,  
refuel probe is retracted, cb external wing control is pushed,  
cb internal wing transfer is pushed, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

l: look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
internal wing transfer is normal, external transfer is off,  
external wing transfer is outbd, wing transfer pressure is normal,  
wing dump is normal, refuel selection is int only,  
landing gear is up, ram air turbine is retracted,  
boost pump left is normal, boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
generators is on, refuel probe is retracted,  
cb external wing control is pushed, cb internal wing transfer is pushed,  
cb fuel valve power is pushed, cb left fuel valve power is pushed,  
cb wing transfer control is pushed, cb left boost pump control is pushed,  
cb boost pump emer control is pushed, cb right boost pump control is pulled,  
cb fuel boost pump is pushed, and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)  
|: look\_at fuel\_quantity\_sector.  
OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      internal wing transfer is normal,  
external transfer is off,      external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,      landing gear is up,  
ram air turbine is retracted,      boost pump left is normal,  
boost pump right is normal,      external store jettison center is norm,  
external store jettison wing is norm,      generators is on,  
refuel probe is retracted,      cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
cb right boost pump control is pulled,      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)  
|: look\_at fuel\_quantity\_counter.  
OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
internal wing transfer is normal,      external transfer is off,  
external wing transfer is outbd,      wing transfer pressure is normal,  
wing dump is normal,      refuel selection is int only,  
landing gear is up,      ram air turbine is retracted,  
boost pump left is normal,      boost pump right is normal,  
external store jettison center is norm,      external store jettison wing is norm,

generators is on,                      refuel probe is retracted,  
cb external wing control is pushed,      cb internal wing transfer is pushed,  
cb fuel valve power is pushed,              cb left fuel valve power is pushed,  
cb wing transfer control is pushed,      cb left boost pump control is pushed,  
cb boost pump emer control is pushed,      cb right boost pump control is pulled,  
cb fuel boost pump is pushed,              and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at,              fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,              internal wing transfer is normal,  
external transfer is off,                  external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,              landing gear is up,  
ram air turbine is retracted,              boost pump left is normal,  
boost pump right is normal,              external store jettison center is norm,  
external store jettison wing is norm,      generators is on,  
refuel probe is retracted,              cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
cb right boost pump control is pulled,      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: identify\_problem.

OK.

Reverse transfer of fuselage fuel caused by:



### A failed open defuel valve

\*\*\*\*\*

The following facts are now true:

reverse transfer fail is problem,      problem is identified,  
external tank is looked at,      fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,      internal wing transfer is normal,  
external transfer is off,      external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,      landing gear is up,  
ram air turbine is retracted,      boost pump left is normal,  
boost pump right is normal,      external store jettison center is norm,  
external store jettison wing is norm,      generators is on,  
refuel probe is retracted,      cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
cb right boost pump control is pulled,      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
type set SWITCH POSITION for status(SWITCH,POSITION)

l: set ram\_air\_turbine extended.

OK.

\*\*\*\*\*

The following facts are now true:

ram air turbine is extended,      reverse transfer fail is problem,  
problem is identified,      external tank is looked at,  
fuel quantity counter is looked at,      fuel quantity sector is looked at,  
centertank jettison is looked at,      centerline tank is looked at,  
internal wing transfer is normal,      external transfer is off,  
external wing transfer is outbd,      wing transfer pressure is normal,  
wing dump is normal,      refuel selection is int only,  
landing gear is up,      boost pump left is normal,  
boost pump right is normal,      external store jettison center is norm,



external store jettison wing is norm, generators is on,  
refuel probe is retracted, cb external wing control is pushed,  
cb internal wing transfer is pushed, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set generators off.

OK.

\*\*\*\*\*

The following facts are now true:

generators is off, ram air turbine is extended,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, internal wing transfer is normal,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is normal, wing dump is normal,  
refuel selection is int only, landing gear is up,  
boost pump left is normal, boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
refuel probe is retracted, cb external wing control is pushed,  
cb internal wing transfer is pushed, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pushed,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set cb\_left\_boost\_pump\_control pulled.

OK.

\*\*\*\*\*

The following facts are now true:

cb left boost pump control is pulled, generators is off,  
ram air turbine is extended, reverse transfer fail is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
internal wing transfer is normal, external transfer is off,  
external wing transfer is outbd, wing transfer pressure is normal,  
wing dump is normal, refuel selection is int only,  
landing gear is up, boost pump left is normal,  
boost pump right is normal, external store jettison center is norm,  
external store jettison wing is norm, refuel probe is retracted,  
cb external wing control is pushed, cb internal wing transfer is pushed,  
cb fuel valve power is pushed, cb left fuel valve power is pushed,  
cb wing transfer control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

```
type look_at INDICATOR for looked_at(INDICATOR)
type set SWITCH POSITION for status(SWITCH,POSITION)
|: set cb_boost_pump_emer_control pulled.
```

OK.

\*\*\*\*\*

The following facts are now true:

cb boost pump emer control is pulled, cb left boost pump control is pulled,  
generators is off, ram air turbine is extended,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, internal wing transfer is normal,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is normal, wing dump is normal,  
refuel selection is int only, landing gear is up,

boost pump left is normal,                    boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
refuel probe is retracted,                    cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb right boost pump control is pulled,   cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set generators on.

OK.

\*\*\*\*\*

The following facts are now true:

generators is on,                    cb boost pump emer control is pulled,  
cb left boost pump control is pulled,   ram air turbine is extended,  
reverse transfer fail is problem,      problem is identified,  
external tank is looked at,           fuel quantity counter is looked at,  
fuel quantity sector is looked at,      centertank jettison is looked at,  
centerline tank is looked at,          internal wing transfer is normal,  
external transfer is off,              external wing transfer is outbd,  
wing transfer pressure is normal,      wing dump is normal,  
refuel selection is int only,          landing gear is up,  
boost pump left is normal,            boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
refuel probe is retracted,            cb external wing control is pushed,  
cb internal wing transfer is pushed,   cb fuel valve power is pushed,  
cb left fuel valve power is pushed,   cb wing transfer control is pushed,  
cb right boost pump control is pulled,   cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : set ram\_air\_turbine retracted.



OK.

\*\*\*\*\*

The following facts are now true:

ram air turbine is retracted, generators is on,  
cb boost pump emer control is pulled, cb left boost pump control is pulled,  
reverse transfer fail is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, internal wing transfer is normal,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is normal, wing dump is normal,  
refuel selection is int only, landing gear is up,  
boost pump left is normal, boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
refuel probe is retracted, cb external wing control is pushed,  
cb internal wing transfer is pushed, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pushed,  
cb right boost pump control is pulled, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

l: set wing\_transfer\_pressure ovr\_d\_trans.

OK.

\*\*\*\*\*

The following facts are now true:

wing transfer pressure is ovr\_d\_trans, ram air turbine is retracted,  
generators is on, cb boost pump emer control is pulled,  
cb left boost pump control is pulled, reverse transfer fail is problem,  
problem is identified, external tank is looked at,  
fuel quantity counter is looked at, fuel quantity sector is looked at,  
centertank jettison is looked at, centerline tank is looked at,  
internal wing transfer is normal, external transfer is off,  
external wing transfer is outbd, wing dump is normal,  
refuel selection is int only, landing gear is up,



boost pump left is normal,                boost pump right is normal,  
external store jettison center is norm, external store jettison wing is norm,  
refuel probe is retracted,                cb external wing control is pushed,  
cb internal wing transfer is pushed,      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,      cb wing transfer control is pushed,  
cb right boost pump control is pulled,   cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,                look\_at centertank\_jettison,  
look\_at fuel\_quantity\_sector,           look\_at fuel\_quantity\_counter,  
look\_at external\_tank,                identify\_problem,  
ram\_air\_turbine to extended,           generators to off,  
cb\_left\_boost\_pump\_control to pulled,   cb\_boost\_pump\_emer\_control to pulled,  
generators to on,                ram\_air\_turbine to retracted,  
wing\_transfer\_pressure to ovrdr\_trans,and terminate\_problem  
yes

! ?- statistics.

atom space: 96K (in use: 37204, max. used: 37204)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 3096)

heap: 1000K (in use: 193916, max. used: 276376)

global stack: 1900K (in use: 0, max. used: 1042716)

local stack: 1000K (in use: 300, max. used: 42304)

Runtime: 188.20 sec.

yes

! ?- halt.

[ Prolog execution halted ]

## I. RANDOM SELECTING CASE

C-Prolog version 1.5

! ?- [expert,tutor].

expert consulted 19672 bytes 2.86667 sec.

tutor consulted 38916 bytes 4.28333 sec.

yes

! ?- random.

! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !

You are the air commander on a F-4E fighter.

Fuel system malfunction occurred and you have to fix the problem.

Your objectives:

problem is identified and problem is terminated.

Wait a moment while I analyze the problem thoroughly.

Type h for help.

\*\*\*\*\*

The following facts are now true:

internal wing transfer is stop trans, external transfer is off,  
external wing transfer is outbd, wing transfer pressure is ovrdrans,  
refuel selection is all tanks, ram air turbine is retracted,  
boost pump left is check, boost pump right is normal,  
generators is on, refuel probe is retracted,  
cb external wing control is pulled, cb internal wing transfer is pulled,  
cb fuel valve power is pushed, cb left fuel valve power is pushed,  
cb wing transfer control is pulled, cb left boost pump control is pushed,  
cb boost pump emer control is pushed, cb right boost pump control is pushed,  
cb fuel boost pump is pushed, and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at centerline\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

centerline tank is looked at, internal wing transfer is stop trans,  
external transfer is off, external wing transfer is outbd,

wing transfer pressure is ovrd trans, refuel selection is all tanks,  
 ram air turbine is retracted, boost pump left is check,  
 boost pump right is normal, generators is on,  
 refuel probe is retracted, cb external wing control is pulled,  
 cb internal wing transfer is pulled, cb fuel valve power is pushed,  
 cb left fuel valve power is pushed, cb wing transfer control is pulled,  
 cb left boost pump control is pushed, cb boost pump emer control is pushed,  
 cb right boost pump control is pushed, cb fuel boost pump is pushed,  
 and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at centertank\_jettison.

OK.

\*\*\*\*\*

The following facts are now true:

centertank jettison is looked at, centerline tank is looked at,  
 internal wing transfer is stop trans, external transfer is off,  
 external wing transfer is outbd, wing transfer pressure is ovrd trans,  
 refuel selection is all tanks, ram air turbine is retracted,  
 boost pump left is check, boost pump right is normal,  
 generators is on, refuel probe is retracted,  
 cb external wing control is pulled, cb internal wing transfer is pulled,  
 cb fuel valve power is pushed, cb left fuel valve power is pushed,  
 cb wing transfer control is pulled, cb left boost pump control is pushed,  
 cb boost pump emer control is pushed, cb right boost pump control is pushed,  
 cb fuel boost pump is pushed, and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_sector.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity sector is looked at,      centertank jettison is looked at,  
 centerline tank is looked at,      internal wing transfer is stop trans,  
 external transfer is off,      external wing transfer is outbd,  
 wing transfer pressure is ovrdrans,      refuel selection is all tanks,  
 ram air turbine is retracted,      boost pump left is check,  
 boost pump right is normal,      generators is on,  
 refuel probe is retracted,      cb external wing control is pulled,  
 cb internal wing transfer is pulled,      cb fuel valve power is pushed,  
 cb left fuel valve power is pushed,      cb wing transfer control is pulled,  
 cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
 cb right boost pump control is pushed,      cb fuel boost pump is pushed,  
 and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
 type set SWITCH POSITION for status(SWITCH,POSITION)

! : look\_at fuel\_quantity\_counter.

OK.

\*\*\*\*\*

The following facts are now true:

fuel quantity counter is looked at,      fuel quantity sector is looked at,  
 centertank jettison is looked at,      centerline tank is looked at,  
 internal wing transfer is stop trans,      external transfer is off,  
 external wing transfer is outbd,      wing transfer pressure is ovrdrans,  
 refuel selection is all tanks,      ram air turbine is retracted,  
 boost pump left is check,      boost pump right is normal,  
 generators is on,      refuel probe is retracted,  
 cb external wing control is pulled,      cb internal wing transfer is pulled,  
 cb fuel valve power is pushed,      cb left fuel valve power is pushed,  
 cb wing transfer control is pulled,      cb left boost pump control is pushed,  
 cb boost pump emer control is pushed,      cb right boost pump control is pushed,  
 cb fuel boost pump is pushed,      and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)  
 type set SWITCH POSITION for status(SWITCH,POSITION)



l: look\_at external\_tank.

OK.

\*\*\*\*\*

The following facts are now true:

external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, internal wing transfer is stop trans,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is ovrd trans, refuel selection is all tanks,  
ram air turbine is retracted, boost pump left is check,  
boost pump right is normal, generators is on,  
refuel probe is retracted, cb external wing control is pulled,  
cb internal wing transfer is pulled, cb fuel valve power is pushed,  
cb left fuel valve power is pushed, cb wing transfer control is pulled,  
cb left boost pump control is pushed, cb boost pump emer control is pushed,  
cb right boost pump control is pushed, cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

l: identify\_problem.

OK.

Wing fuel leak problem caused by:

Mechanical problem

\*\*\*\*\*

The following facts are now true:

wing fuel leaking is problem, problem is identified,  
external tank is looked at, fuel quantity counter is looked at,  
fuel quantity sector is looked at, centertank jettison is looked at,  
centerline tank is looked at, internal wing transfer is stop trans,  
external transfer is off, external wing transfer is outbd,  
wing transfer pressure is ovrd trans, refuel selection is all tanks,  
ram air turbine is retracted, boost pump left is check,  
boost pump right is normal, generators is on,  
refuel probe is retracted, cb external wing control is pulled,

cb internal wing transfer is pulled,    cb fuel valve power is pushed,  
cb left fuel valve power is pushed,    cb wing transfer control is pulled,  
cb left boost pump control is pushed,    cb boost pump emer control is pushed,  
cb right boost pump control is pushed,    cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : disengage afterburner.

OK.

\*\*\*\*\*

The following facts are now true:

disengage is afterburner,                      wing fuel leaking is problem,  
problem is identified,                      external tank is looked at,  
fuel quantity counter is looked at,    fuel quantity sector is looked at,  
centertank jettison is looked at,    centerline tank is looked at,  
internal wing transfer is stop trans,    external transfer is off,  
external wing transfer is outbd,    wing transfer pressure is ovrdr trans,  
refuel selection is all tanks,    ram air turbine is retracted,  
boost pump left is check,    boost pump right is normal,  
generators is on,    refuel probe is retracted,  
cb external wing control is pulled,    cb internal wing transfer is pulled,  
cb fuel valve power is pushed,    cb left fuel valve power is pushed,  
cb wing transfer control is pulled,    cb left boost pump control is pushed,  
cb boost pump emer control is pushed,    cb right boost pump control is pushed,  
cb fuel boost pump is pushed,    and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

! : fly below\_threehundredsfts.

OK.

\*\*\*\*\*

The following facts are now true:

below threehundredsfts is flight,    disengage is afterburner,

wing fuel leaking is problem,                      problem is identified,  
external tank is looked at,                      fuel quantity counter is looked at,  
fuel quantity sector is looked at,                      centertank jettison is looked at,  
centerline tank is looked at,                      internal wing transfer is stop trans,  
external transfer is off,                      external wing transfer is outbd,  
wing transfer pressure is ovrd trans,                      refuel selection is all tanks,  
ram air turbine is retracted,                      boost pump left is check,  
boost pump right is normal,                      generators is on,  
refuel probe is retracted,                      cb external wing control is pulled,  
cb internal wing transfer is pulled,                      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,                      cb wing transfer control is pulled,  
cb left boost pump control is pushed,                      cb boost pump emer control is pushed,  
cb right boost pump control is pushed,                      cb fuel boost pump is pushed,  
and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: set refuel\_probe refuel.

OK.

\*\*\*\*\*

The following facts are now true:

refuel probe is refuel,                      below threehundredskits is flight,  
disengage is afterburner,                      wing fuel leaking is problem,  
problem is identified,                      external tank is looked at,  
fuel quantity counter is looked at,                      fuel quantity sector is looked at,  
centertank jettison is looked at,                      centerline tank is looked at,  
internal wing transfer is stop trans,                      external transfer is off,  
external wing transfer is outbd,                      wing transfer pressure is ovrd trans,  
refuel selection is all tanks,                      ram air turbine is retracted,  
boost pump left is check,                      boost pump right is normal,  
generators is on,                      cb external wing control is pulled,  
cb internal wing transfer is pulled,                      cb fuel valve power is pushed,  
cb left fuel valve power is pushed,                      cb wing transfer control is pulled,  
cb left boost pump control is pushed,                      cb boost pump emer control is pushed,  
cb right boost pump control is pushed,                      cb fuel boost pump is pushed,



and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: land as \_soon\_ as \_practicable.

OK.

\*\*\*\*\*

The following facts are now true:

as soon as practicable is landing,	refuel probe is refuel,
below threehundredskts is flight,	disengage is afterburner,
wing fuel leaking is problem,	problem is identified,
external tank is looked at,	fuel quantity counter is looked at,
fuel quantity sector is looked at,	centertank jettison is looked at,
centerline tank is looked at,	internal wing transfer is stop trans,
external transfer is off,	external wing transfer is outbd,
wing transfer pressure is ovrdr trans,	refuel selection is all tanks,
ram air turbine is retracted,	boost pump left is check,
boost pump right is normal,	generators is on,
cb external wing control is pulled,	cb internal wing transfer is pulled,
cb fuel valve power is pushed,	cb left fuel valve power is pushed,
cb wing transfer control is pulled,	cb left boost pump control is pushed,
cb boost pump emer control is pushed,	cb right boost pump control is pushed,
cb fuel boost pump is pushed,	and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: land arrested.

OK.

\*\*\*\*\*

The following facts are now true:

arrested is landing,	as soon as practicable is landing,
refuel probe is refuel,	below threehundredskts is flight,
disengage is afterburner,	wing fuel leaking is problem,
problem is identified,	external tank is looked at,



fuel quantity counter is looked at,      fuel quantity sector is looked at,  
 centertank jettison is looked at,      centerline tank is looked at,  
 internal wing transfer is stop trans,      external transfer is off,  
 external wing transfer is outbd,      wing transfer pressure is ovrd trans,  
 refuel selection is all tanks,      ram air turbine is retracted,  
 boost pump left is check,      boost pump right is normal,  
 generators is on,      cb external wing control is pulled,  
 cb internal wing transfer is pulled,      cb fuel valve power is pushed,  
 cb left fuel valve power is pushed,      cb wing transfer control is pulled,  
 cb left boost pump control is pushed,      cb boost pump emer control is pushed,  
 cb right boost pump control is pushed,      cb fuel boost pump is pushed,  
 and cb refuel probe is pushed.

What operator do you choose?

For example:

type look\_at INDICATOR for looked\_at(INDICATOR)

type set SWITCH POSITION for status(SWITCH,POSITION)

|: terminate\_problem.

OK.

Full list of your actions:

look\_at centerline\_tank,      look\_at centertank\_jettison,  
 look\_at fuel\_quantity\_sector,      look\_at fuel\_quantity\_counter,  
 look\_at external\_tank,      identify\_problem,  
 disengage afterburner,      fly below\_threehundredsfts,  
 refuel\_probe to refuel,      land as\_soon\_as\_practicable,  
 land arrested,      and terminate\_problem

yes

| ?- statistics.

atom space: 96K (in use: 37088, max. used: 37088)

aux. stack: 8K (in use: 0, max. used: 600)

trail: 64K (in use: 48, max. used: 3660)

heap: 1000K (in use: 148884, max. used: 208936)

global stack: 1900K (in use: 0, max. used: 737624)

local stack: 1000K (in use: 300, max. used: 43472)

Runtime: 104.13 sec.

yes

| ?- halt.

## APPENDIX B

### SOURCE CODE OF EXPERT SYSTEM

```

/*****
/* Title : EXPERT */
/* Author: KANG, Moung-Hung */
/* Date : 21 May 1990 */
/* Description : This program provides expert knowledge of fuel system in */
/* F-4 aircraft to implement means-ends analysis. Program consists */
/* three parts; the definition of switch and switch position, the */
/* the components of means-ends analysis, and the definition of */
/* problem to send current and goal states to the tutor. */

/* How to run : */
/* 1. Load 'expert' and 'tutor' program.*/
/* 2. Enter the execution command to run the specific */
/*      emergency, as like below. */
/*      internal : internal wing fuel fails to transfer case */
/*      center   : centerline fuel fails to transfer case */
/*      external : external wing fuel fails to transfer case */
/*      reverse  : reverse transfer of fuselage fuel case */
/*      stream   : fuel streaming from underside of aircraft */
/*      following center jettison case */
/*      leaking  : wing fuel leaking case */
/*      normal   : normal case */
/*      general  : reverse transfer of fuselage fuel using */
/*      total switch states */
/*      random   : select current states by using random number */
/* 3. If the user want debugflag, use 'asserta(debugflag)' */
*****/

intro('! WELCOME TO PILOT EMERGENCY TUTORING SYSTEM !
      You are the air commander on a F-4E fighter. Fuel system malfunction occurred
      and you have to fix the problem.').
```

/\* Available switch positions to handle fuel system malfunction \*/

position(internal\_wing\_transfer,normal,stop\_trans).  
position(external\_transfer,off,outbd).  
position(wing\_transfer\_pressure,normal,ovrd\_trans).  
position(wing\_dump,normal,dump).  
position(refuel\_selection,int\_only,all\_tanks).  
position(landing\_gear,up,down).  
position(ram\_air\_turbine,extended,retracted).  
position(boost\_pump\_left,check,normal).  
position(boost\_pump\_right,check,normal).  
position(external\_store\_jettison\_center,norm,jett).  
position(external\_store\_jettison\_wing,norm,jett).  
position(generators,on,off).  
position(refuel\_probe,retracted,extended,refuel).  
position(external\_wing\_transfer,outbd,wing,center).  
position(cb\_external\_wing\_control,pushed,pulled).  
position(cb\_internal\_wing\_transfer,pushed,pulled).  
position(cb\_fuel\_valve\_power,pushed,pulled).  
position(cb\_left\_fuel\_valve\_power,pushed,pulled).  
position(cb\_wing\_transfer\_control,pushed,pulled).  
position(cb\_left\_boost\_pump\_control,pushed,pulled).  
position(cb\_right\_boost\_pump\_control,pushed,pulled).  
position(cb\_boost\_pump\_emer\_control,pushed,pulled).  
position(cb\_fuel\_boost\_pump,pushed,pulled).  
position(cb\_refuel\_probe,pushed,pulled).

/\* Available indicator values to determine the problem \*/

indication(fuel\_quantity\_sector,decrease,normal,increase).  
indication(fuel\_quantity\_counter,decrease,increase,severe\_decrea).  
indication(external\_tank,empty,not\_empty,full).  
indication(centerline\_tank,empty,not\_empty).  
indication(center\_tank\_jettison,jettisoned,not\_jettisoned).

```

/* Definition of the switch and indicator */
switch(SW):- binary_switch(SW); tri_switch(SW).
binary_switch(SW):- position(SW,X,Y).
tri_switch(SW):- position(SW,X,Y,Z).

existposition(SW,X):- binary_switch(SW),
    (position(SW,X,Y); position(SW,Y,X)).
existposition(SW,X):- tri_switch(SW),
    (position(SW,X,Y,Z); position(SW,Y,X,Z); position(SW,Y,Z,X)).

find_opposite(SW,X,Y):- binary_switch(SW),
    (position(SW,X,Y); position(SW,Y,X)).

indica(IND):- binary_indica(IND); tri_indica(IND).
binary_indica(IND):- indication(IND,X,Y).
tri_indica(IND):- indication(IND,X,Y,Z).

existindication(IND,X):- binary_indica(IND),
    (indication(IND,X,Y); indication(IND,Y,X)).
existindication(IND,X):- tri_indica(IND),
    (indication(IND,X,Y,Z); indication(IND,Z,X,Y); indication(IND,Y,Z,X)).

/* Recommended operator for fuel system malfunction */

recommended([identified(problem)],identify_problem).
recommended([terminated(problem)],terminate_problem).
recommended([flight(below_threehundredsfts)],fly(below_threehundredsfts)).
recommended([flight(disengage,afterburner)],disengage(afterburner)).
recommended([landing(as_soon_as_practicable)],land(as_soon_as_practicable)).
recommended([landing(arrested)],land(arrested)).

recommended([status(SW,X)],set(SW,X)):- switch(SW),existposition(SW,X).
recommended([looked_at(IND)],look_at(IND)):- indica(IND).

```



/\* Precondition for internal wing fuel fails to transfer \*/

```
precondition(identify_problem,_,  
    [looked_at(fuel_quantity_sector),  
    looked_at(fuel_quantity_counter),  
    looked_at(external_tank),  
    looked_at(centerline_tank),  
    looked_at(centertank_jettison)]).
```

```
precondition(set(internal_wing_transfer,normal),  
    [problem(internal_wing_transfer_fail)],  
    [identified(problem),  
    status(internal_wing_transfer,stop_trans),  
    status(external_transfer,off)]).
```

```
precondition(set(refuel_probe,retracted),  
    [problem(internal_wing_transfer_fail)],  
    [identified(problem),  
    status(external_transfer,off),  
    status(internal_wing_transfer,normal),  
    not status(refuel_probe,retracted)]).
```

```
precondition(set(cb_internal_wing_transfer,pushed),  
    [problem(internal_wing_transfer_fail)],  
    [identified(problem),  
    status(cb_internal_wing_transfer,pulled),  
    status(refuel_probe,retracted),  
    status(external_transfer,off),  
    status(internal_wing_transfer,normal)]).
```

```
precondition(set(wing_transfer_pressure,ovrd_trans),  
    [problem(internal_wing_transfer_fail)],  
    [identified(problem),  
    status(wing_transfer_pressure,normal),  
    status(external_transfer,off),  
    status(internal_wing_transfer,normal),
```

```
status(refuel_probe,retracted),
status(cb_internal_wing_transfer,pushed))].
```

```
precondition(terminate_problem,
[problem(internal_wing_transfer_fail)],
[identified(problem),
status(external_transfer,off),
status(internal_wing_transfer,normal),
status(refuel_probe,retracted),
status(cb_internal_wing_transfer,pushed),
status(wing_transfer_pressure,ovrd_trans)]).
```

```
/* Precondition for centerline fuel fails to transfer */
precondition(set(cb_external_wing_control,pushed),
[problem(centerline_transfer_fail)], [identified(problem),
status(external_wing_transfer,center),
status(cb_external_wing_control,pulled)]).
```

```
precondition(set(cb_left_fuel_valve_power,pushed),
[problem(centerline_transfer_fail)],
[identified(problem),
status(external_wing_transfer,center),
status(cb_external_wing_control,pushed),
status(cb_left_fuel_valve_power,pulled)]).
```

```
precondition(set(refuel_probe,retracted),
[problem(centerline_transfer_fail)],
[identified(problem),
status(external_wing_transfer,center),
status(cb_external_wing_control,pushed),
status(cb_left_fuel_valve_power,pushed),
not status(refuel_probe,retracted)]).
```

```
precondition(set(wing_transfer_pressure,ovrd_trans),
[problem(centerline_transfer_fail)],
[identified(problem),
```

```
status(external_wing_transfer,center),
status(cb_external_wing_control,pushed),
status(cb_left_fuel_valve_power,pushed),
status(refuel_probe,retracted),
status(wing_transfer_pressure,normal)))).
```

```
precondition(terminate_problem,
[problem(centerline_transfer_fail)],
[identified(problem),
status(external_wing_transfer,center),
status(cb_external_wing_control,pushed),
status(cb_left_fuel_valve_power,pushed),
status(refuel_probe,retracted),
status(wing_transfer_pressure,ovrd_trans)]).
```

/\* Precondition for external wing fuel fails to transfer \*/

```
precondition(set(cb_external_wing_control,pushed),
[problem(external_wing_transfer_fail)],
[identified(problem),
status(external_wing_transfer,outbd),
status(cb_external_wing_control,pulled)]).
```

```
precondition(set(refuel_probe,retracted),
[problem(external_wing_transfer_fail)],
[identified(problem),
status(external_wing_transfer,outbd),
status(cb_external_wing_control,pushed),
not status(refuel_probe,retracted)]).
```

```
precondition(set(cb_fuel_valve_power,pushed),
[problem(external_wing_transfer_fail)],
[identified(problem),
status(external_wing_transfer,outbd),
status(cb_external_wing_control,pushed),
status(refuel_probe,retracted),
status(cb_fuel_valve_power,pulled)]).
```

```

precondition(set(wing_transfer_pressure,ovrd_trans),
  [problem(external_wing_transfer_fail)],
  [identified(problem),
    status(external_wing_transfer,outbd),
    status(cb_external_wing_control,pushed),
    status(refuel_probe,retracted),
    status(cb_fuel_valve_power,pushed),
    status(wing_transfer_pressure,normal)]).

```

```

precondition(terminate_problem,
  [problem(external_wing_transfer_fail)],
  [identified(problem),
    status(external_wing_transfer,outbd),
    status(cb_external_wing_control,pushed),
    status(refuel_probe,retracted),
    status(cb_fuel_valve_power,pushed),
    status(wing_transfer_pressure,ovrd_trans)]).

```

```

/* Precondition for reverse transfer of fuselage fuel */
precondition(set(generators,off),
  [problem(reverse_transfer_fail)],
  [identified(problem),
    status(generators,on),
    status(ram_air_turbine,extended)]).

```

```

precondition(set(cb_wing_transfer_control,pushed),
  [problem(reverse_transfer_fail)],
  [identified(problem),
    status(cb_wing_transfer_control,pulled),
    status(ram_air_turbine,extended),
    status(generators,off)]).

```

```

precondition(set(cb_left_boost_pump_control,pulled),
  [problem(reverse_transfer_fail)],
  [identified(problem),
    status(cb_left_boost_pump_control,pushed),

```



status(ram\_air\_turbine,extended),  
status(generators,off),  
status(cb\_wing\_transfer\_control,pushed))].

precondition(set(cb\_boost\_pump\_emer\_control,pulled),  
[problem(reverse\_transfer\_fail)],  
[identified(problem),  
status(cb\_boost\_pump\_emer\_control,pushed),  
status(ram\_air\_turbine,extended),  
status(generators,off),  
status(cb\_wing\_transfer\_control,pushed),  
status(cb\_left\_boost\_pump\_control,pulled)]]).

precondition(set(cb\_right\_boost\_pump\_control,pulled),  
[problem(reverse\_transfer\_fail)],  
[identified(problem),  
status(cb\_right\_boost\_pump\_control,pushed),  
status(ram\_air\_turbine,extended),  
status(generators,off),  
status(cb\_wing\_transfer\_control,pushed),  
status(cb\_left\_boost\_pump\_control,pulled),  
status(cb\_boost\_pump\_emer\_control,pulled)]]).

precondition(set(generators,on),  
[problem(reverse\_transfer\_fail)],  
[identified(problem),  
status(generators,off),  
status(ram\_air\_turbine,extended),  
status(cb\_wing\_transfer\_control,pushed),  
status(cb\_left\_boost\_pump\_control,pulled),  
status(cb\_boost\_pump\_emer\_control,pulled),  
status(cb\_right\_boost\_pump\_control,pulled)]]).

precondition(set(ram\_air\_turbine,retracted),  
[problem(reverse\_transfer\_fail)],  
[identified(problem),

```

status(ram_air_turbine,extended),
status(generators,on),
status(cb_wing_transfer_control,pushed),
status(cb_right_boost_pump_control,pulled),
status(cb_left_boost_pump_control,pulled),
status(cb_boost_pump_emer_control,pulled)]];

```

```

precondition(set(wing_transfer_pressure,ovrd_trans),
[problem(reverse_transfer_fail)],
[identified(problem),
status(wing_transfer_pressure,normal),
status(cb_wing_transfer_control,pushed),
status(cb_left_boost_pump_control,pulled),
status(cb_boost_pump_emer_control,pulled),
status(cb_right_boost_pump_control,pulled),
status(generators,on),
status(ram_air_turbine,retracted)]]).

```

```

precondition(terminate_problem,
[problem(reverse_transfer_fail)],
[identified(problem),
status(ram_air_turbine,retracted),
status(generators,on),
status(cb_wing_transfer_control,pushed),
status(cb_left_boost_pump_control,pulled),
status(cb_boost_pump_emer_control,pulled),
status(cb_right_boost_pump_control,pulled),
status(wing_transfer_pressure,ovrd_trans)]]).

```

/\* Fuel streaming from underside of aircraft following center jettision \*/

```

precondition(set(cb_refuel_probe,pulled),
[problem(fuel_streaming_underside)],
[identified(problem),
status(wing_transfer_pressure,normal),
status(cb_refuel_probe,pushed)]]).

```

```
precondition(set(refuel_probe,refuel),
  [problem(fuel_streaming_underside)],
  [identified(problem),
    status(wing_transfer_pressure,normal),
    status(cb_refuel_probe,pulled),
    not status(refuel_probe,refuel)]).
```

```
precondition(set(cb_fuel_boost_pump,pulled),
  [problem(fuel_streaming_underside)],
  [identified(problem),
    status(wing_transfer_pressure,normal),
    status(cb_refuel_probe,pulled),
    status(refuel_probe,refuel),
    status(cb_fuel_boost_pump,pushed)]).
```

```
precondition(terminate_problem,
  [problem(fuel_streaming_underside)],
  [identified(problem),
    status(wing_transfer_pressure,normal),
    status(cb_refuel_probe,pulled),
    status(refuel_probe,refuel),
    status(cb_fuel_boost_pump,pulled)]).
```

/\* Precondition for Wing fuel leak \*/

```
precondition(disengage(afterburner),_,[identified(problem)]).
```

```
precondition(fly(below_threehundredsfts),
  [problem(wing_fuel_leaking)],
  [identified(problem),
    flight(disengage,afterburner)]).
```

```
precondition(set(refuel_probe,refuel),
  [problem(wing_fuel_leaking)],
  [identified(problem),
    flight(disengage,afterburner),
```

```
flight(below_threehundredsfts),
not status(refuel_probe,refuel)]).
```

```
precondition(land(as_soon_as_practicable),
[problem(wing_fuel_leaking)],
[identified(problem),
flight(disengage,afterburner),
flight(below_threehundredsfts),
status(refuel_probe,refuel)]).
```

```
precondition(land(arrested),
[problem(wing_fuel_leaking)],
[flight(disengage,afterburner),
flight(below_threehundredsfts),
status(refuel_probe,refuel),
landing(as_soon_as_practicable)]).
```

```
precondition(terminate_problem,
[problem(wing_fuel_leaking)],
[flight(disengage,afterburner),
flight(below_threehundredsfts),
status(refuel_probe,refuel),
landing(as_soon_as_practicable),
landing(arrested)]).
```

```
/* Generic precondition */
```

```
precondition(set(SW,X),_,[status(SW,OX),identified(problem)]):-
    binary_switch(SW), find_opposite(SW,X,OX).
precondition(set(SW,X),_,[identified(problem), not status(SW,X)]):-
    tri_switch(SW).
precondition(look_at(IND),_,[]):- indica(IND).
```

```
/* Deletepostcondition for fuel system malfunction */
```

```
deletepostcondition(identify_problem,_,
[indicator(fuel_quantity_sector,V),
```



```
indicator(fuel_quantity_counter,W),  
indicator(centerline_tank,X),  
indicator(external_tank,Y)]]).
```

```
deletepostcondition(terminate_problem,  
    [problem(internal_wing_transfer_fail)],  
    [problem(internal_wing_transfer_fail)]).
```

```
deletepostcondition(terminate_problem,  
    [problem(centerline_transfer_fail)],  
    [problem(centerline_transfer_fail)]).
```

```
deletepostcondition(terminate_problem,  
    [problem(external_wing_transfer_fail)],  
    [problem(external_wing_transfer_fail)]).
```

```
deletepostcondition(terminate_problem,  
    [problem(reverse_transfer_fail)],  
    [problem(reverse_transfer_fail)]).
```

```
deletepostcondition(terminate_problem,  
    [problem(fuel_streaming_underside)],  
    [problem(fuel_streaming_underside)]).
```

```
deletepostcondition(terminate_problem,  
    [problem(wing_fuel_leaking)],  
    [problem(wing_fuel_leaking)]).
```

```
/* Deletepostcondition of triple switch */  
deletepostcondition(set(external_wing_transfer,outbd),_,  
    [status(external_wing_transfer,wing),  
    status(external_wing_transfer,center)]).
```

```
deletepostcondition(set(external_wing_transfer,wing),_,  
    [status(external_wing_transfer,center),  
    status(external_wing_transfer,outbd)]).
```

```
deletepostcondition(set(external_wing_transfer,center),_,  
    [status(external_wing_transfer,outbd),  
    status(external_wing_transfer,wing)]).
```

```
deletepostcondition(set(refuel_probe,retracted),_,  
    [status(refuel_probe,extended),  
    status(refuel_probe,refuel)]).
```

```
deletepostcondition(set(refuel_probe,extended),_,  
    [status(refuel_probe,refuel),  
    status(refuel_probe,retracted)]).
```

```
deletepostcondition(set(refuel_probe,refuel),_,  
    [status(refuel_probe,retracted),  
    status(refuel_probe,extended)]).
```

```
deletepostcondition(disengage(afterburner),_,[]).  
deletepostcondition(fly(below_threehundredsfts),_,[]).  
deletepostcondition(land(as_soon_as_practicable),_,[]).  
deletepostcondition(land(arrested),_,[]).
```

```
/* Generic deletepostcondition */  
deletepostcondition(set(SW,X),_,[status(SW,OX)]):-  
    binary_switch(SW),find_opposite(SW,X,OX).
```

```
deletepostcondition(look_at(IND),_,[]):- indica(IND).
```

```
/* Addpostcondition for fuel system malfunction */
```

```
addpostcondition(identify_problem,  
    [indicator(fuel_quantity_sector,decrease),  
    indicator(fuel_quantity_counter,decrease),  
    indicator(external_tank,empty),  
    indicator(centerline_tank,empty)],  
    [problem(internal_wing_transfer_fail), identified(problem)]).
```

'Failure of internal wing fuel to transfer caused by:

1. The wing tanks failing to pressurize
2. The wing transfer valves failing to open').

```
addpostcondition(identify_problem,  
  [indicator(fuel_quantity_sector,normal),  
   indicator(fuel_quantity_counter,decrease),  
   indicator(external_tank,full),  
   indicator(centerline_tank,not_empty)],  
  [problem(centerline_transfer_fail), identified(problem)],  
  'Failure of the centerline fuel to transfer caused by:  
  1. The defueling shutoff valve failing to open position  
  2. The refueling valve failing to the closed position  
  3. The tank failing to become pressurized').
```

```
addpostcondition(identify_problem,  
  [indicator(fuel_quantity_sector,normal),  
   indicator(fuel_quantity_counter,decrease),  
   indicator(external_tank,not_empty),  
   indicator(centerline_tank,empty)],  
  [problem(external_wing_transfer_fail), identified(problem)],  
  'Failure of the external wing fuel to transfer caused by:  
  1. The external wing shutoff valve failing to the closed  
  2. The tanks failing to become pressurized').
```

```
addpostcondition(identify_problem,  
  [indicator(fuel_quantity_sector,increase),  
   indicator(fuel_quantity_counter,increase),  
   indicator(external_tank,not_empty),  
   indicator(centerline_tank,empty),  
   not indicator(center_tank_jettison,jettisoned)],  
  [problem(reverse_transfer_fail), identified(problem)],  
  'Reverse transfer of fuselage fuel caused by:  
  A failed open defuel valve').
```

```

addpostcondition(identify_problem,
    [indicator(fuel_quantity_sector,increase),
    indicator(fuel_quantity_counter,increase),
    indicator(external_tank,full),
    indicator(centerline_tank,not_empty),
    not indicator(center_tank_jettison,jettisoned)],
    [problem(reverse_transfer_fail), identified(problem)],
    'Reverse transfer of fuselage fuel caused by:
    A failed open defuel valve').

```

```

addpostcondition(identify_problem,
    [indicator(center_tank_jettison,jettisoned),
    indicator(fuel_quantity_sector,increase),
    indicator(fuel_quantity_counter,increase),
    indicator(external_tank,not_empty),
    indicator(centerline_tank,empty)],
    [problem(fuel_streaming_underside), identified(problem)],
    'Fuel streaming from underside of aircraft following CL jett
    1. Cockpit indications are identical to reverse transfer
    2. Fuel loss on the sector and counter: 500~700 lbs/min').

```

```

addpostcondition(identify_problem,
    [indicator(fuel_quantity_sector,normal),
    indicator(fuel_quantity_counter,severe_decrea),
    indicator(external_tank,not_empty),
    indicator(centerline_tank,empty)],
    [problem(wing_fuel_leaking), identified(problem)],
    'Wing fuel leak problem caused by: Mechanical problem').

```

```

addpostcondition(identify_problem,_,
    [identified(problem),terminated(problem)],
    'There is no problem.').

```

```

addpostcondition(disengage(afterburner),_,
    [flight(disengage,afterburner)]).

```



```

addpostcondition(fly(below_threehundredsfts),
    [problem(wing_fuel_leaking)], [flight(below_threehundredsfts)]).
addpostcondition(land(as_soon_as_practicable),_,
    [landing(as_soon_as_practicable)]).
addpostcondition(land(arrested),_,
    [landing(arrested)]).

```

```

/* Generic addpostcondition */

```

```

addpostcondition(set(SW,X),_,[status(SW,X)]):-
    switch(SW),existposition(SW,X).

```

```

addpostcondition(look_at(IND),_,[looked_at(IND)]):-
    indica(IND),existindication(IND,X).
addpostcondition(terminate_problem,_,[terminated(problem)]).

```

```

/* Problem for internal wing fuel fails to transfer */

```

```

internal:- tutor([indicator(fuel_quantity_sector,decrease),
    indicator(fuel_quantity_counter,decrease),
    indicator(external_tank,empty),
    indicator(centerline_tank,empty),
    indicator(center_tank_jettison,not_jettisoned),
    status(external_transfer,off),
    status(internal_wing_transfer,stop_trans),
    status(refuel_probe,extended),
    status(cb_internal_wing_transfer,pulled),
    status(wing_transfer_pressure,normal)],
    [identified(problem),terminated(problem)]).

```

```

/* Problem for centerline fuel fails to transfer */

```

```

center:- tutor([indicator(fuel_quantity_sector,normal),
    indicator(fuel_quantity_counter,decrease),
    indicator(external_tank,full),
    indicator(centerline_tank,not_empty),
    indicator(center_tank_jettison,not_jettisoned),
    status(external_wing_transfer,wing),
    status(cb_external_wing_control,pulled),

```

```
status(cb_left_fuel_valve_power,pulled),
status(refuel_probe,extended),
status(wing_transfer_pressure,normal)],
[identified(problem),terminated(problem)]]).
```

/\* Problem for external wing fuel fails to transfer \*/

```
external:- tutor([indicator(fuel_quantity_sector,normal),
indicator(fuel_quantity_counter,decrease),
indicator(external_tank,not_empty),
indicator(centerline_tank,empty),
indicator(center_tank_jettison,not_jettisoned),
status(external_wing_transfer,center),
status(cb_external_wing_control,pulled),
status(refuel_probe,extended),
status(cb_fuel_valve_power,pulled),
status(wing_transfer_pressure,normal)],
[identified(problem),terminated(problem)]]).
```

/\* Problem for reverse transfer of fuselage fuel \*/

```
reverse:- tutor([indicator(fuel_quantity_sector,increase),
indicator(fuel_quantity_counter,increase),
indicator(external_tank,not_empty),
indicator(centerline_tank,empty),
indicator(center_tank_jettison,not_jettisoned),
status(ram_air_turbine,retracted),
status(generators,on),
status(cb_wing_transfer_control,pushed),
status(cb_left_boost_pump_control,pushed),
status(cb_boost_pump_einer_control,pushed),
status(cb_right_boost_pump_control,pulled),
status(wing_transfer_pressure,normal)],
[identified(problem),terminated(problem)]]).
```

/\* Problem for fuel streaming from underside of aircraft following jettison \*/

```
stream:- tutor([indicator(center_tank_jettison,jettisoned),
indicator(fuel_quantity_sector,increase),
```

```

indicator(fuel_quantity_counter,increase),
indicator(external_tank,not_empty),
indicator(centerline_tank,empty),
status(wing_transfer_pressure,ovrd_trans),
status(cb_refuel_probe,pushed),
status(refuel_probe,extended),
status(cb_fuel_boost_pump,pushed)],
[identified(problem),terminated(problem)]).

```

/\* Problem for wing fuel leak \*/

```

leaking:- tutor([indicator(fuel_quantity_sector,normal),
indicator(fuel_quantity_counter,severe_decrea),
indicator(external_tank,not_empty),
indicator(centerline_tank,empty),
indicator(centertank_jettison,not_jettisoned),
status(refuel_probe,retracted)],
[identified(problem),terminated(problem)]).

```

/\* Test of normal case \*/

```

normal:- tutor([indicator(fuel_quantity_sector,increase),
indicator(fuel_quantity_counter,decrease),
indicator(external_tank,not_empty),
indicator(centerline_tank,empty),
indicator(centertank_jettison,not_jettisoned),
status(internal_wing_transfer,normal),
status(external_transfer,off),
status(external_wing_transfer,outbd),
status(wing_transfer_pressure,normal),
status(wing_dump,normal),
status(refuel_selection,int_only),
status(landing_gear,up),
status(ram_air_turbine,retracted),
status(boost_pump_left,normal),
status(boost_pump_right,normal),
status(external_store_jettison_center,norm),
status(external_store_jettision_wing,norm),

```

```

status(generators,on),
status(refuel_probe,retracted),
status(cb_external_wing_control,pushed),
status(cb_internal_wing_transfer,pushed),
status(cb_fuel_valve_power,pushed),
status(cb_left_fuel_valve_power,pushed),
status(cb_wing_transfer_control,pushed),
status(cb_left_boost_pump_control,pushed),
status(cb_boost_pump_emer_control,pushed),
status(cb_right_boost_pump_control,pulled),
status(cb_fuel_boost_pump,pushed),
status(cb_refuel_probe,pushed)],
[identified(problem),terminated(problem)]]).

```

/\* Problem for reverse transfer of fuselage fuel using total status \*/

```

general:- tutor([indicator(fuel_quantity_sector,increase),
    indicator(fuel_quantity_counter,increase),
    indicator(external_tank,not_empty),
    indicator(centerline_tank,empty),
    indicator(center_tank_jettison,not_jettisoned),
    status(internal_wing_transfer,normal),
    status(external_transfer,off),
    status(external_wing_transfer,outbd),
    status(wing_transfer_pressure,normal),
    status(wing_dump,normal),
    status(refuel_selection,int_only),
    status(landing_gear,up),
    status(ram_air_turbine,retracted),
    status(boost_pump_left,normal),
    status(boost_pump_right,normal),
    status(external_store_jettison_center,norm),
    status(external_store_jettison_wing,norm),
    status(generators,on),
    status(refuel_probe,retracted),
    status(cb_external_wing_control,pushed),
    status(cb_internal_wing_transfer,pushed),

```



```

status(cb_fuel_valve_power,pushed),
status(cb_left_fuel_valve_power,pushed),
status(cb_wing_transfer_control,pushed),
status(cb_left_boost_pump_control,pushed),
status(cb_boost_pump_emer_control,pushed),
status(cb_right_boost_pump_control,pulled),
status(cb_fuel_boost_pump,pushed),
status(cb_refuel_probe,pushed)],
[identified(problem),terminated(problem)]).

```

/\* Problem for random selecting indicator and switch states \*/

```

random:- randinit, randitem([decrease,normal,increase],I1),
        randitem([decrease,increase,severe_decrea],I2),
        randitem([empty,not_empty,full],I3),
        randitem([empty,not_empty],I4),
        randitem([jettisoned,not_jettisoned],I5),
        randitem([normal,stop_trans],S1),
        randitem([off,outbd],S2),
        randitem([outbd,wing,center],S3),
        randitem([normal,ovrd_trans],S4),
        randitem([int_only,all_tanks],S5),
        randitem([extended,retracted],S6),
        randitem([check,normal],S7),
        randitem([check,normal],S8),
        randitem([on,off],S9),
        randitem([retracted,extended,refuel],S10),
        randitem([pushed,pulled],S11),
        randitem([pushed,pulled],S12),
        randitem([pushed,pulled],S13),
        randitem([pushed,pulled],S14),
        randitem([pushed,pulled],S15),
        randitem([pushed,pulled],S16),
        randitem([pushed,pulled],S17),
        randitem([pushed,pulled],S18),
        randitem([pushed,pulled],S19),
        randitem([pushed,pulled],S20),

```

```

tutor([indicator(fuel_quantity_sector,I1),
      indicator(fuel_quantity_counter,I2),
      indicator(external_tank,I3),
      indicator(centerline_tank,I4),
      indicator(center_tank_jettison,I5),
      status(internal_wing_transfer,S1),
      status(external_transfer,S2),
      status(external_wing_transfer,S3),
      status(wing_transfer_pressure,S4),
      status(refuel_selection,S5),
      status(ram_air_turbine,S6),
      status(boost_pump_left,S7),
      status(boost_pump_right,S8),
      status(generators,S9),
      status(refuel_probe,S10),
      status(cb_external_wing_control,S11),
      status(cb_internal_wing_transfer,S12),
      status(cb_fuel_valve_power,S13),
      status(cb_left_fuel_valve_power,S14),
      status(cb_wing_transfer_control,S15),
      status(cb_left_boost_pump_control,S16),
      status(cb_boost_pump_emer_control,S17),
      status(cb_right_boost_pump_control,S18),
      status(cb_fuel_boost_pump,S19),
      status(cb_refuel_probe,S20)],
      [identified(problem),terminated(problem)]).

```

## APPENDIX C

### SOURCE CODE OF TUTORING SYSTEM

```
/* Title : TUTOR */
/* Author : Neil, C. Rowe */
/* Modify : Kang, Moungh-Hung */
/* Date : 21 Jan 1990 */
/* Description : Problem-independent code for 'means-ends tutoring': */
/* tutoring for learning of sequences modelable by means-ends analysis. */
/* Underline means modified program with original program. */

/* This is the top level of the means-ends tutor. */
tutor(STATE,GOAL) :- not check_obvious_errors, issue_warnings, do_intro,
    write('Your objectives: '), nl, writelist(GOAL,state), write(' '), nl, randinit,
    uniqueassert(top_goal(GOAL)), find_operators(XL),
    uniqueassert(op_list(XL)), write('Wait a moment while I analyze the problem
    thoroughly. '), nl, once_means_ends(STATE,GOAL,OPLIST2,GOALSTATE2),
    uniqueassert(top_solution(OPLIST2)), abolish(mainline_states,4),
    write('Type h for help. '), nl,
    means_ends_tutor(STATE,GOAL,OPLIST,GOALSTATE,[],[]), nl,
    write('Full list of your actions: '), nl, writelist(OPLIST,op), nl, !.
tutor(STATE,GOAL) :- write('Too bad: a solution is now impossible. '), nl, !.

means_ends_tutor(STATE,GOAL,[],STATE,STACK,GOALSTACK) :-
    difference(GOAL,STATE,[],) !.
means_ends_tutor(STATE,GOAL,OPLIST,STATE,STACK,GOALSTACK) :-
    member([STATE,GOAL],STACK), !, fail.
means_ends_tutor(STATE,GOAL,OPLIST,STATE,STACK,GOALSTACK) :-
    not once_means_ends(STATE,GOAL,OPLIST,GOALSTATE), !, fail.
means_ends_tutor(STATE,GOAL,OPLIST,GOALSTATE,STACK,GOALSTACK) :-
    difference(GOAL,STATE,D), applicable_op(D,OP),
    get_precondition(OP,STATE,PRELIST), all_achievable(STATE,PRELIST),
    apply_op(OP,STATE,STATE2),
```

```

once_means_ends(STATE2,GOAL,OPLIST2,GOALSTATE2), !,
means_ends_tutor(STATE,PRELIST,PREOPLIST,PRESTATE,
[[STATE,GOAL]STACK], [GOAL|GOALSTACK]), !,
met(STATE,GOAL,OPLIST,GOALSTATE,STACK,PRELIST,PREOPLIST,
PRESTATE,OP,D, GOALSTACK).

met(STATE,GOAL,PREOPLIST,PRESTATE,STACK,PRELIST,PREOPLIST,
PRESTATE,OP,D,GOALSTACK) :- difference(GOAL,PRESTATE,[]), !.
met(STATE,GOAL,PREOPLIST,PRESTATE,STACK,PRELIST,PREOPLIST,
PRESTATE,OP,D,GOALSTACK) :-
higher_goal_achieved(GOALSTACK,PRESTATE), !.
met(STATE,GOAL,OPLIST,GOALSTATE,STACK,PRELIST,PREOPLIST,
PRESTATE,OP,D, GOALSTACK) :- difference(GOAL,PRESTATE,D2),
not_applicable_op(D2,OP), !,
means_ends_tutor(PRESTATE,GOAL,OPLIST2,GOALSTATE,[],
GOALSTACK), append(PREOPLIST,OPLIST2,OPLIST).
met(STATE,GOAL,OPLIST,GOALSTATE,STACK,PRELIST,PREOPLIST,
PRESTATE,OP,D, GOALSTACK) :- writedebug8(OP),
check_with_student(OP,PRESTATE,D,NEWOP),
get_deletepostcondition(NEWOP,PRESTATE,DELETEPOSTLIST),
print_optional_message_d(NEWOP,PRESTATE),
deleteitems(DELETEPOSTLIST,PRESTATE,PRESTATE2),
get_addpostcondition(NEWOP,PRESTATE,ADDPOSTLIST),
print_optional_message_a(NEWOP,PRESTATE),
union(ADDPOSTLIST,PRESTATE2,POSTLIST2),
do_randsubst(NEWOP,POSTLIST2,POSTLIST),
check_mainline_return(POSTLIST), !,
means_ends_tutor(POSTLIST,GOAL,POSTOPLIST,GOALSTATE,[],[GOAL|
GOALSTACK]), append(PREOPLIST,[NEWOP|POSTOPLIST],OPLIST).

do_intro :- intro(T), write(T), nl, !.
do_intro.

find_operators(XL) :- nice_bagof(X,P^recommended(P,X),XL).

```



```

/* Problem-definition errors */
/*that is, errors by the instructor building a particular means_ends tutor.*/

check_obvious_errors :- setof([M,A],obvious_error(M,A),MAL), !,
    writepairlist(MAL).

obvious_error('precondition fact missing for operator ',O) :-
    recommended(D,O), not get_precondition(O,S,L).
obvious_error('deletepostcondition fact missing for operator ',O) :-
    recommended(D,O), not get_deletepostcondition(O,S,L).
obvious_error('addpostcondition fact missing for operator ',O) :-
    recommended(D,O), not get_addpostcondition(O,S,L).
obvious_error('recommended fact missing for operator ',O) :-
    get_precondition(O,S,L), not recommended(D,O).
obvious_error('recommended fact missing for operator ',O) :-
    get_deletepostcondition(O,S,L), not recommended(D,O).
obvious_error('recommended fact missing for operator ',O) :-
    get_addpostcondition(O,S,L), not recommended(D,O).

issue_warnings :- setof([M,A],possible_error(M,A),MAL), !, write('Warnings:'),
    nl, writepairlist(MAL), nl.
issue_warnings.

possible_error('This fact is not creatable: ',F) :- get_precondition(O,S,PL),
    backtracking_member(F,PL), uncreatable(F).

writepairlist([]).
writepairlist([[X,Y]|L]) :- write(X), write(Y), nl, writepairlist(L).

/* Handling of randomness */
do_randsubst(O,S,NS) :- randsubst(O,RL), !, do_randsubst2(RL,S,NS).
do_randsubst(O,S,S).

do_randsubst2([],S,S).
do_randsubst2([[F,NF,P]|L],S,NS) :- rand(1000,K), P1000 is P*1000, K=<P1000,
    changestate(F,NF,S,S2), !, do_randsubst2(L,S2,NS).

```

```

do_randsubst2([F,NF,P,M]L],S,NS) :- rand(1000,K), P1000 is P*1000, K=<P1000,
    changestate(F,NF,S,S2), !, write(M), nl, do_randsubst2(L,S2,NS).
do_randsubst2([C]L],S,NS) :- do_randsubst2(L,S,NS).
changestate(none,NF,S,[NF|S]) :- !, not member(NF,S), write('Random change
    made: fact '), writefact(NF,state,Any), write(' added. '), nl, !.
changestate(F,none,S,S2) :- !, member(F,S), delete(F,S,S2), write('Random change
    made: fact '), writefact(NF,state,Any), write(' removed. '), nl, !.
changestate(F,NF,S,[NF|S3]) :- !, member(F,S), delete(F,S,S3), write('Random
    change made: fact '), writefact(NF,state,Any1), write(' added, and '), nl,
    write(' fact '), writefact(F,state,Any2), write(' removed. '), nl, !.

permutation([],[]) :- !.
permutation(L,[I|PL]) :- randitem(L,I), delete(I,L,L2), permutation(L2,PL).

randitem(L,I) :- length(L,N), rand(N,KM1), K is KM1+1, item(K,L,I).
rand(N,K) :- randseed(S), nexttrand(S,NS), K is NS mod N, retract(randseed(S)),
    asserta(randseed(NS)).
nexttrand(S,NS) :- FIC is 100*cputime, IC is floor(FIC), S2 is ((S*S)+IC) mod 2311,
    S3 is (S2*S2) mod 2311, NS is (S3*S) mod 2311, not(S=NS).
nexttrand(S,NS) :- NS is (S+1) mod 2311.
randinit :- T is cputime, get4digits(T,FC), S is FC mod 2311,
    uniqueassert(randseed(S)), !.
randinit :- C is cputime*1000, FC is floor(C), S is FC mod 2311,
    uniqueassert(randseed(S)).
get4digits(S,N) :- name(S,AS), remove_nonnums(AS,[C1,C2,C3,C4,C5,C6|CL]),
    name(N,[C6,C5,C4,C3]).

remove_nonnums([],[]).
remove_nonnums([AX|L],L2) :- AX<48, !, remove_nonnums(L,L2), !.
remove_nonnums([AX|L],[AX|L2]) :- remove_nonnums(L,L2), !.

/* Tutoring rules */
/* This first rule obtains an operator from the student. */
check_with_student(O,S,D,NO) :-
    write('*****'), nl,
    write('The following facts are now true:'), nl,

```

deleteitems([indicator(fuel quantity sector,X),  
indicator(fuel quantity counter,Y),indicator(external tank,Z),  
indicator(centerline tank,V),indicator(center tank jettison,W)],S,DS),  
writelist(DS,state), write(' '), nl, write('What operator do you choose? '), nl,  
write(' For example:'), nl, write(' type look at INDICATOR for  
looked at(INDICATOR)'), nl, write(' type set SWITCH POSITION for  
status(SWITCH.POSITION)'), nl, niceread(AO2), space\_parse(AO2,O3),  
handle\_student\_op(O3,O,S,D,NO).

handle\_student\_op(O,O,S,D,O) :- !, write('OK.'), nl.  
handle\_student\_op(' ',O,S,D,NO) :- !, check\_with\_student(O,S,D,NO).  
handle\_student\_op(O2,O,S,D,NO) :- helpword(O2), !, op\_list(OL),  
permutation(OL,POL), write('The possible operators are: '), nl,  
writelist(POL,op), write(' '), nl, write('Your objectives are: '), nl, top\_goal(G),  
writelist(G,state), write(' '), nl, check\_with\_student(O,S,D,NO).  
handle\_student\_op(O2,O,S,D,O2) :- xnopref(O2,O), !, write('OK.'), nl.  
handle\_student\_op(O2,O,S,D,O) :- op\_list(OL), not singlemember(O2,OL),  
fixspell(O2,O), !, write('I assume you mean '), writefact(O,op,Any), write(' '),  
nl.  
handle\_student\_op(O2,O,S,D,NO) :- op\_list(OL), not singlemember(O2,OL),  
backtracking\_member(O3,OL), fixspell(O2,O3), !, write('I assume you mean '),  
writefact(O3,op,Any), write(' '), nl, handle\_student\_op(O3,O,S,D,NO).  
  
handle\_student\_op(O2,O,S,D,NO) :- op\_list(OL), not singlemember(O2,OL), !,  
write('Not a valid operator--please choose one of: '), nl, permutation(OL,POL),  
writelist(POL,op), write(' '), nl, check\_with\_student(O,S,D,NO).  
  
handle\_student\_op(O2,O,S,D,NO) :- get\_precondition(O2,S,PO2),  
difference(PO2,S,D2), not D2=[], !, write('That operator requires that '), nl,  
writelist(D2,precond), write(' '), nl, check\_with\_student(O,S,D,NO).  
handle\_student\_op(O2,O,S,D,NO) :- apply\_op(O2,S,S), write('That will not affect  
anything.'), nl, check\_with\_student(O,S,D,NO).  
handle\_student\_op(O2,O,S,D,NO) :- apply\_op(O2,S,S2), top\_goal(G), not  
once\_means\_ends(S2,G,OL2,GS2), !, write('You cannot ever succeed if you do  
that.'), nl, check\_with\_student(O,S,D,NO).



```

handle_student_op(O2,O,S,D,O2) :- top_goal(G), apply_op(O,S,S3),
    apply_op(O2,S,S2), compare_solutions(S3,G,OL3,GS3,S2,G,OL2,GS2),
    subsequence([O|OL3],OL2), !, apply_ops([O|OL3],S,SL,GS4),
    elimdups(SL,ESL), asserta(mainline_states(ESL,O2,S,O)),
    write('That does not seem immediately helpful, but I will try it. '), nl.
handle_student_op(O2,O,S,D,O2) :- top_goal(G), once_means_ends(S,G,OL,FS),
    not member(O2,OL), !, write('I will try it, but it is not recommended or needed
    for the problem. '), nl.
handle_student_op(O2,O,S,D,O2) :- top_goal(G), difference(G,S,D2),
    all_achievable(S,D2), applicable_op(D2,O3), get_precondition(O3,PL),
    least_common_op(S,G,O,O2,PL,GROOT), !, write('I will try it, but it is not
    recommended first when '), nl, difference(GROOT,S,D5),
    delete_uncreatable(D5,D6), permutation(D6,D7), writelist(D7,precond),
    write(' '), nl.
handle_student_op(O2,O,S,D,O2) :- write('Not the operator I would choose,
    but let us try it. '), nl, !.

```

/\* Intermediate predicates used by the tutor \*/

```
xnopref(O1,O2) :- nopref(O1,O2).
```

```
xnopref(O1,O2) :- nopref(O2,O1).
```

/\* This is used when the student has picked an operator which does help \*/

/\* solve the problem but is not the highest-priority operator (i.e., \*/

/\* he has a bug in his internal 'recommended' definitions.). \*/

```

least_common_op(S,G,O,O2,G2,G) :- once_means_ends(S,G2,OL,NS),
    least_common_op2(O,O2,OL).

```

```

least_common_op(S,G,O,O2,G2,DROOT) :- difference(G2,S,D),
    all_achievable(S,D), applicable_op(D,O3), get_precondition(O3,S,G3),
    least_common_op(S,G2,O,O2,G3,DROOT), !.

```

```
least_common_op2(O,O2,OL) :- not member(O,OL), !.
```

```
least_common_op2(O,O2,OL) :- not member(O2,OL), !.
```

```

compare_solutions(S3,G,OL3,GS3,S2,G,OL2,GS2) :-
    once_means_ends(S3,G,OL3,GS3), once_means_ends(S2,G,OL2,GS2), !.

```



```

/* Since the tutor repeatedly reexamines slightly different paths to the */
/* goal, a lot of redundancy can be avoided by having the tutor store every */
/* solution it has found (by 'means_ends') to a problem. And fact */
/* order shouldn't matter in caching states. */
cache_states(S,G,[],GS) :- !.
cache_states(S,G,OL,GS) :- cached(S,G,OL,GS), !.
cache_states(S,G,OL,GS) :- cached(S2,G2,OL2,GS2), check_permutation(S,S2),
    check_permutation(G,G2), !.
cache_states(S,G,[O|OL],GS) :- asserta(cached(S,G,[O|OL],GS)),
    apply_op(O,S,NS), cache_states(NS,G,OL,GS), !.

/* This takes a list of operators and tells you what the resulting */
/* state is after applying them to some starting state. */
apply_ops([],S,[S],S) :- !.
apply_ops([O|OL],S,[S|SL],NS) :- apply_op(O,S,S2), apply_ops(OL,S2,SL,NS).
apply_op(O,S,NS) :- get_deletepostcondition(O,S,DP), deleteitems(DP,S,S2),
    get_addpostcondition(O,S,AP), union(AP,S2,NS), !.

helpword(help).
helpword(h).
helpword(huh).

/* This checks for when the student returns from a digression, so as */
/* to tutor him at that point. */
check_mainline_return(S) :- mainline_states(SL,O,OS,BO),
    check_mainline_return2(S,SL,O,OS,BO).
check_mainline_return(S).

check_mainline_return2(S,[S2|SL],O,OS,BO) :- permutemember(S,[S2]), !,
    write('You are returning to a previous state. '), nl.
check_mainline_return2(S,SL,O,OS,BO) :- permutemember(S,SL), !,
    write('Do you see now that your choice of the '), writefact(O,op,Any1),
    write(' action in the state with the facts ['), writelist(OS,state),
    write('] was not the best choice; the '), writefact(BO,op,Any2),
    write(' action would have been better. '), nl,
    retract(mainline_states(SL,O,OS,BO)).

```

```

higher_goal_achieved(GL,S) :- higher_goal_achieved2(GL,S).
higher_goal_achieved2([],S) :- !, fail.
higher_goal_achieved2([G|GL],S) :- difference(G,S,[]), !.
higher_goal_achieved2([G|GL],S) :- higher_goal_achieved2(GL,S).

```

```

confusable(O,O,S) :- !, fail.

```

```

/* Two actions are confusable if they are opposites */

```

```

confusable(O1,O2,S) :- get_deletepostcondition(O1,S,DL1),
    get_deletepostcondition(O2,S,DL2), add_deletepostcondition(O1,S,DL2),
    add_deletepostcondition(O2,S,DL1), !.

```

```

/* Or if their first word is identical */

```

```

confusable(O1,O2,S) :- O1=..[P|R1], O2=..[P|R2], !.

```

```

/* Or if they're words whose first two letters are identical */

```

```

confusable(O1,O2,S) :- atom(O1), atom(O2), name(O1,[C1,C2|NO1]),
    name(O2,[C1,C2|NO2]), !.

```

```

/* This part is modified for natural language output */

```

```

writelist([],R) :- !.

```

```

writelist([X],R) :- !, writefact(X,R,TL).

```

```

writelist([X,Y],R) :- !, writefact(X,R,TL1), write(' and '), writefact(Y,R,TL2).

```

```

writelist(L,R) :- writelist2(L,R,1,0).

```

```

writelist2([X],R,I,TL) :- checkodd(I), !, write('and '), writefact(X,R,NL).

```

```

writelist2([X],R,I,TL) :- not checkodd(I), TL<34, !, SL is TL+4, writeblank(SL),
    write('and '), writefact(X,R,NL), !.

```

```

writelist2([X],R,I,TL) :- not checkodd(I), !, write('and '), writefact(X,R,NL).

```

```

writelist2([X|L],R,I,TL) :- checkodd(I), writefact(X,R,NL), write(','), NI = I+1,
    writelist2(L,R,NI,NL).

```

```

writelist2([X|L],R,I,TL) :- not checkodd(I), writeblank(TL), writefact(X,R,NL),
    write(','), nl, NI = I+1, writelist2(L,R,NI,0).

```

/\* utility predicate \*/

checkodd(1).

checkodd(N) :- N>1, N1 is N-2, checkodd(N1).

writeblank(N) :- N < 40, WN is 40-N, writeblank1(WN).

writeblank(N) :- N >= 40, write(' ').

writeblank1(0) :- !.

writeblank1(N) :- write(' '), NN is N-1, writeblank1(NN).

writefact(not F,state,TL) :- atom(F), !, name(F,LF), space length(LF,LLF,SLF),  
name(SF,SLF), write('it is not '), write(SF), TL is LLF+10, !.

writefact(not F,state,TL) :- F=..[P,X], atom(X), !, name(P,PL), name(X,XL),  
space length(PL,LPL,SPL), space length(XL,LXL,SXL), name(SX,SXL),  
name(SP,SPL), write(SX), is form(X,IX), write(IX), write('not '), write(SP), !,  
TL1 is LPL+LXL, TL is TL1+8.

writefact(not F,state,TL) :- F=..[status,X,Y], !, name(X,XL), name(Y,YL),  
space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),  
name(SY,SYL), write(SX), is form(Y,IY), write(IY), write('not '), write(SY),  
TL1 is LXL+LYL, TL is TL1+8, !.

writefact(not F,state,TL) :- F=..[indicator,X,Y], !, name(X,XL), name(Y,YL),  
space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),  
name(SY,SYL), write(SX), is form(Y,IY), write(IY), write('not '), write(SY),  
TL1 is LXL+LYL, TL is TL1+8, !.

writefact(not F,state,TL) :- F=..[P,X,Y], !, name(X,XL), name(Y,YL),  
space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),  
name(SY,SYL), write(SX), write(' '), is form(Y,IY), write(IY), write('not '),  
write(SY), TL1 is LXL+LYL, TL is TL1+7, !.

writefact(F,state,TL) :- atom(F), !, name(F,LF), space length(LF,LLF,SLF),  
name(SF,SLF), write('it is '), write(SF), TL is LLF+6, !.

writefact(F,state,TL) :- F=..[P,X], atom(X), !, name(P,PL), name(X,XL),  
space length(PL,LPL,SPL), space length(XL,LXL,SXL), name(SP,SPL),  
name(SX,SXL), write(SX), is form(X,IX), write(IX), write(SP), TL1 is  
LPL+LXL, TL is TL1+4, !.

writefact(F,state,TL) :- F=..[status,X,Y], !, name(X,XL), name(Y,YL),  
space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),



```

name(SY,SYL), write(SX), is form(Y,IY), write(IY), write(SY),
TL1 is LXL+LYL, TL is TL1+4, !.
writefact(F,state,TL) :- F=..[indicator,X,Y], !, name(X,XL), name(Y,YL),
    space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),
name(SY,SYL), write(SX), is form(Y,IY), write(IY), write(SY),
TL1 is LXL+LYL, TL is TL1+4, !.
writefact(F,state,TL) :- F=..[P,X,Y], !, name(X,XL), name(Y,YL),
    space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),
name(SY,SYL), write(SX), write(' '), is form(Y,IY), write(IY), write(SY),
TL1 is LXL+LYL, TL is TL1+4, !.

/* writefact for preconditions */

writefact(not F,precond,TL) :- atom(F), !, name(F,FL), space length(FL,LFL,SFL),
    name(SF,SFL), write('it is not '), write(SF), TL is LFL+11, !.
writefact(not F,precond,TL) :- F=..[P,X], atom(X), !, name(X,XL), name(P,PL),
    space length(XL,LXL,SXL), space length(PL,LPL,SPL), name(SX,SXL),
name(SP,SPL), write(SX), write(' is not '), write(SP), TL1 is LXL+LPL,
TL is TL1+8, !.
writefact(not F,precond,TL) :- F=..[P,X,Y], !, name(X,XL), name(Y,YL),
    space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),
name(SY,SYL), write(SY), write(' is not '), write(SX), TL1 is LXL+LYL,
TL is TL1+8, !.

writefact(F,precond,TL) :- atom(F), !, name(F,FL), space length(FL,LFL,SFL),
    name(SF,SFL), write('it is '), write(SF), TL is LFL+7.
writefact(F,precond,TL) :- F=..[P,X], atom(X), !, name(X,XL), name(P,PL),
    space length(XL,LXL,SXL), space length(PL,LPL,SPL), name(SX,SXL),
name(SP,SPL), write(SX), is form(X,IX), write(IX), write(SP),
TL1 is LXL+LPL, TL is TL1+4, !.
writefact(F,precond,TL) :- F=..[status,X,Y], name(X,XL), name(Y,YL),
    space length(XL,LXL,SXL), space length(YL,LYL,SYL), name(SX,SXL),
name(SY,SYL), write(SX), is form(Y,IY), write(IY), write(SY),
TL1 is LXL+LYL, TL is TL1+4, !.

```



```

writefact(F,precond,TL) :- F=..[indicator,X,Y], name(X,XL), name(Y,YL),
    space_length(XL,LXL,SXL), space_length(YL,LYL,SYL), name(SX,SXL),
    name(SY,SYL), write(SX), is form(Y,IY), write(IY), write(SY),
    TL1 is LXL+LYL, TL is TL1+4, !.
writefact(F,precond,TL) :- F=..[P,X,Y], name(X,XL), name(Y,YL),
    space_length(XL,LXL,SXL), space_length(YL,LYL,SYL), name(SX,SXL),
    name(SY,SYL), write(SX), write(' '), is form(Y,IY), write(IY), write(SY),
    TL1 is LXL+LYL, TL is TL1+3, !.
writefact(F,op,TL) :- atom(F), !, name(F,FL), length(FL,LFL), write(F),
    TL is LFL+7, !.
writefact(F,op,TL) :- F=..[P,A], name(P,PL), name(A,AL), length(PL,LPL),
    length(AL,LAL), write(P), write(' '), write(A), TL1 is LPL+LAL, TL is TL1+1, !.
writefact(F,op,TL) :- F=..[set,A,B], name(A,AL), name(B,BL), length(AL,LAL),
    length(BL,LBL), write(A), write(' to '), write(B), TL1 is LAL+LBL,
    TL is TL1+4, !.
writefact(F,op,TL) :- F=..[P,A,B], name(A,AL), name(B,BL), length(AL,LAL),
    length(BL,LBL), write(A), write(' to '), write(B), TL1 is LAL+LBL,
    TL is TL1+4, !.
writefact(F,op,TL) :- name(F,FL), length(FL,TL), write(F), !.
writefact(F,R,TL) :- name(F,FL), length(FL,TL), write(F), !.

```

```

is form(X,' is ') :- not atom(X), !.
is form(X,' is ') :- name(X,NX), last(NX,115), !.
is form(X,' is ').

```

/\* The original means-ends program (used for 'what if' reasoning) \*/

```

once_means_ends(STATE,GOAL,OPLIST,GOALSTATE) :-
    means_ends(STATE,GOAL,OPLIST,GOALSTATE),
    cache_states(STATE,GOAL,OPLIST,GOALSTATE), !.

means_ends(STATE,GOAL,OPLIST,GOALSTATE) :-
    means_ends2(STATE,GOAL,OPLIST,GOALSTATE,[]), writedebug7.

means_ends2(STATE,GOAL,OPLIST,GOALSTATE,STACK) :-
    cached(STATE2,GOAL2,OPLIST,GOALSTATE),

```

```

    check_permutation(GOAL,GOAL2), check_permutation(STATE,STATE2), !,
    writedebug6(STACK), !.
means_ends2(STATE,GOAL,OPLIST,GOALSTATE,STACK) :-
    member([STATE,GOAL],STACK), !, writedebug4(STATE,GOAL,STACK), fail.
means_ends2(STATE,GOAL,[],STATE,STACK) :- difference(GOAL,STATE,[],), !.
means_ends2(STATE,GOAL,OPLIST,GOALSTATE,STACK) :-
    difference(GOAL,STATE,D), applicable_op(D,OPERATOR),
    get_precondition(OPERATOR,STATE,PRELIST),
    all_achievable(STATE,PRELIST), writedebug1(D,OPERATOR,STACK),
    means_ends2(STATE,PRELIST,PREOPLIST,PRESTATE,
    [[STATE,GOAL]|STACK]), writedebug2(PRESTATE,D,OPERATOR,STACK),
    get_deletepostcondition(OPERATOR,PRESTATE,DELETEPOSTLIST),
    deleteitems(DELETEPOSTLIST,PRESTATE,PRESTATE2),
    get_addpostcondition(OPERATOR,PRESTATE,ADDPOSTLIST),
    union(ADDPOSTLIST,PRESTATE2,POSTLIST),
    means_ends2(POSTLIST,GOAL,POSTOPLIST,GOALSTATE,
    [[STATE,GOAL]|STACK]), writedebug3(GOALSTATE,OPERATOR,STACK),
    append(PREOPLIST,[OPERATOR|POSTOPLIST],OPLIST).
means_ends2(STATE,GOAL,OPLIST,GOALSTATE,STACK) :-
    writedebug5(STATE,GOAL,STACK), !, fail.

/* Debugging tools */
writedebug1(D,O,STACK) :- not debugflag, !.
writedebug1(D,O,STACK) :- length(STACK,NM1), N is NM1+1,
    write('>>Operator '), write(O), write(' suggested at level '), write(N), nl,
    write('to achieve difference of :'), nl, writelist(D,state), nl, !.
writedebug2(S,D,O,STACK) :- not debugflag, !.
writedebug2(S,D,O,STACK) :- length(STACK,NM1), N is NM1+1,
    write('>>Operator '), write(O), write(' applied at level '), write(N), nl,
    write('to reduce difference of :'), nl, writelist(D,state), nl,
    write('in state in which :'), nl, writelist(S,state), nl, !.

writedebug3(S,O,STACK) :- not debugflag, !.
writedebug3(S,O,STACK) :- length(STACK,NM1), N is NM1+1, write('>>Level '),
    write(N), write(' terminated at state in which :'), nl, writelist(S,state), nl, !.

```

```

writedebug4(S,G,STACK) :- not debugflag, !.
writedebug4(S,G,STACK) :- write('>>>>Reasoning avoided an infinite loop at
    level '), length(STACK,NM1), N is NM1+1, write(N), write(' where problem
    was identical to that at level '), index([S,G],STACK,I), write(I), nl, !.

writedebug5(STATE,GOAL,STACK) :- not debugflag, !.
writedebug5(STATE,GOAL,STACK) :- write('>>>>Unsolvable problem at level '),
    length(STACK,NM1), N is NM1+1, write(N), nl, write('for state :'), nl,
    writelist(STATE,state), nl, write('and goal :'), nl, writelist(GOAL,state), nl, !.

writedebug6(STACK) :- not debugflag, !.
writedebug6(STACK) :- write('>>>>Previously computed solution used at level '),
    length(STACK,NM1), N is NM1+1, write(N), nl, !.

writedebug7 :- not debugflag, !.
writedebug7 :- nl, !.

writedebug8(OP) :- not debugflag, !.
writedebug8(OP) :- write('The tutor prefers operator :'),
    writefact(OP,op,Any), nl, !.

/* Miscellaneous utility functions */
delete_uncreatable([],[]).
delete_uncreatable([X|L],M) :- uncreatable(X), !, delete_uncreatable(L,M).
delete_uncreatable([X|L],[X|M]) :- delete_uncreatable(L,M).
all_achievable(S,G) :- difference(G,S,D), not unachievable_member(D).

unachievable_member(D) :- backtracking_member(F,D), uncreatable(F).

uncreatable(F) :- get_precondition(O,S,L), backtracking_member(F,L),
    not in_postcondition(F).

in_postcondition(not F) :- any_deletepostcondition(O,DPL), member(F,DPL), !.
in_postcondition(not F) :- randsubst(O,RSL), member([F,X,Y,Z],RSL), !.
in_postcondition(not F) :- randsubst(O,RSL), member([F,X,Y],RSL), !.

```



```

in_postcondition(F) :- not F=..[not,P], any_addpostcondition(O,APL),
    member(F,APL), !.
in_postcondition(F) :- not F=..[not,P], randsubst(O,RSL),
    member([X,F,Y,Z],RSL), !.
in_postcondition(F) :- not F=..[not,P], randsubst(O,RSL),
    member([X,F,Y],RSL), !.

any_deletepostcondition(O,L) :- deletepostcondition(O,C,L).
any_deletepostcondition(O,L) :- deletepostcondition(O,L).
any_addpostcondition(O,L) :- addpostcondition(O,C,L).
any_addpostcondition(O,L) :- addpostcondition(O,L).

get_deletepostcondition(O,S,L) :- deletepostcondition(O,C,L,M), factssubset(C,S), !.
get_deletepostcondition(O,S,L) :- deletepostcondition(O,C,L), factssubset(C,S), !.
get_deletepostcondition(O,S,L) :- deletepostcondition(O,L).
get_addpostcondition(O,S,L) :- addpostcondition(O,C,L,M), factssubset(C,S), !.
get_addpostcondition(O,S,L) :- addpostcondition(O,C,L), factssubset(C,S), !.
get_addpostcondition(O,S,L) :- addpostcondition(O,L).
get_precondition(O,S,L) :- precondition(O,C,L,M), factssubset(C,S), !.
get_precondition(O,S,L) :- precondition(O,C,L), factssubset(C,S), !.
get_precondition(O,S,L) :- precondition(O,L).

print_optional_message_d(O,S) :- deletepostcondition(O,C,L,M), factssubset(C,S),
    write(M), nl, !.
print_optional_message_d(O,S) :- !.
print_optional_message_a(O,S) :- addpostcondition(O,C,L,M), factssubset(C,S),
    write(M), nl, !.
print_optional_message_a(O,S) :- !.

applicable_op(D,O) :- recommended(D2,O), subset(D2,D).

/* Spelling correction */
fixspell(W1,W2) :- atom(W1), atom(W2), !, name(W1,AW1),
    fixspell2(AW1,AW2), name(W2,AW2).
fixspell(W1,W2) :- W1=..[P1L], W2=..[P2L], not(P1=P2), !, fixspell(P1,P2).
fixspell(W1,W2) :- W1=..[P,Q1L], W2=..[P,Q2L], not(Q1=Q2), !, fixspell(Q1,Q2).

```



```
fixspell(W1,W2) :- W1=..[P,Q,R1|L], W2=..[P,Q,R2|L], not(R1=R2), !,  
    fixspell(R1,R2).
```

```
fixspell2(AW,AW2) :- deleteone(X,AW,AW2).  
fixspell2(AW,AW2) :- deleteone(X,AW2,AW).  
fixspell2(AW,AW2) :- transpose(AW,AW2).
```

```
transpose([X,Y|L],[Y,X|L]).  
transpose([X|L],[X|M]) :- transpose(L,M).
```

```
/* List utilities */
```

```
deleteone(X,[X|L],L).  
deleteone(X,[Y|L],[Y|M]) :- deleteone(X,L,M).
```

```
difference([],S,[]).  
difference([not P|G],S,G2) :- not singlemember(P,S), !, difference(G,S,G2).  
difference([P|G],S,G2) :- singlemember(P,S), !, difference(G,S,G2).  
difference([P|G],S,[P|G2]) :- difference(G,S,G2).
```

```
subset([],L).  
subset([X|L],L2) :- singlemember(X,L2), subset(L,L2).
```

```
factsubset([],L).  
factsubset([not P|L],L2) :- not singlemember(P,L2), !, factsubset(L,L2).  
factsubset([not P|L],L2) :- !, fail.  
factsubset([P|L],L2) :- singlemember(P,L2), factsubset(L,L2).
```

```
member(X,L) :- singlemember(X,L).
```

```
singlemember(X,[X|L]) :- !.  
singlemember(X,[Y|L]) :- singlemember(X,L).  
append([],L,L).  
append([X|L],L2,[X|M3]) :- append(L,L2,L3).
```

```

union([],L,L).
union([X|L1],L2,L3) :- singlemember(X,L2), !, union(L1,L2,L3).
union([X|L1],L2,[X|L3]) :- union(L1,L2,L3).

deleteitems([],L,L).
deleteitems([X|L],L2,L3) :- delete(X,L2,L4), deleteitems(L,L4,L3).

delete(X,[],[]).
delete(X,[X|L],M) :- !, delete(X,L,M).
delete(X,[Y|L],[Y|M]) :- delete(X,L,M).

item(K,[],I) :- !, fail.
item(K,[X|L],X) :- K=<1, !.
item(K,[X|L],Y) :- KM1 is K-1, item(KM1,L,Y).

check_permutation(L,M) :- subset(L,M), subset(M,L), !.

subsequence([],L) :- !.
subsequence([X|L],[X|M]) :- !, subsequence(L,M).
subsequence(L,[X|M]) :- subsequence(L,M).

permutemember(X,[X|L]) :- !.
permutemember(X,[Y|L]) :- subset(X,Y), subset(Y,X), !.
permutemember(X,[Y|L]) :- permutemember(X,L).

last([X],X). last([X|L],Y) :- last(L,Y).

elimdups([],[]).
elimdups([X|L],M) :- singlemember(X,L), !, elimdups(L,M).
elimdups([X|L],[X|M]) :- elimdups(L,M).

uniqueassert(Q) :- retract(Q), !, asserta(Q).
uniqueassert(Q) :- asserta(Q).

backtracking_member(X,[X|L]).
backtracking_member(X,[Y|L]) :- backtracking_member(X,L).

```

```

/* I/O stuff */
space_parse([32|AS1],S1) :- space_parse(AS1,S1), !.
space_parse(AS1,S1) :- not member(32,AS1), remove_ugly_chars(AS1,AS2),
    name(S1,AS2), !.
space_parse(AS1,S2) :- remove_ugly_chars(AS1,AS2), append(L1,[32|L2],AS2),
    name(N1,L1), parse_args(L2,N2), S2=..[N1|N2].

parse_args([],[]) :- !.
parse_args([32|AL],L) :- parse_args(AL,L), !.
parse_args(AL,[L]) :- not member(32,AL), name(L,AL), !.
parse_args(AL,[N1|N2]) :- append(L1,[32|L2],AL), name(N1,L1),
    parse_args(L2,N2), !.

remove_ugly_chars([],[]).
remove_ugly_chars([X|L],M) :- X<65, not X=32, !, remove_ugly_chars(L,M).
remove_ugly_chars([X|L],[X|M]) :- remove_ugly_chars(L,M).

niceread(L) :- checkretract(readbuff(L2)), asserta(readbuff([])), niceread2(L), !.

niceread2(L) :- get0(C), niceread3(C,L).

niceread3(10,L) :- !, readbuff(L2), reverse(L2,L).
niceread3(C,L) :- readbuff(L3), retract(readbuff(L3)), asserta(readbuff([C|L3])),
    niceread2(L).

checkretract(S) :- call(S), retract(S), !.
checkretract(S).

reverse(L,R) :- reverse2(L,[],R).

reverse2([],L,L) :- !.
reverse2([X|L],R,S) :- reverse2(L,[X|R],S).

index(X,[X|L],1) :- !.
index(X,[Y|L],N) :- index(X,L,Nm1), N is Nm1+1.

```

```
nice_bagof(X,P,L) :- bagof(X,P,L), !.  
nice_bagof(X,P,[]).
```

```
/* Modified underscore -> space in length program */
```

```
space_length([],0,[]).
```

```
space_length([_5|L1],N1,[_2|L2]) :- space_length(L1,N2,L2), N1 is N2+1.
```

```
space_length([X|L1],N1,[X|L2]) :- space_length(L1,N2,L2), N1 is N2+1.
```



## LIST OF REFERENCES

1. Paul E. Lehner, *Artificial Intelligence and National Defense: Opportunity and Challenge*, TAB BOOKS, 1989.
2. Stuart C. Shapiro, "Military Applications," *Encyclopedia of Artificial Intelligence*, p.604-614, 1987.
3. Kenneth J. Stein, "Researchers Channel AI Activities Toward Real-World Application," *Aviation Week & Space Technology*, p. 40, Feb 17, 1986.
4. Decker, W. L., *Application of Artificial Intelligence To Improve Aircraft Survivability*, M.S. Thesis, Naval Postgraduate School, Monterey, California, Dec 1985.
5. *Aviation Week & Space Technology*, p. 118, Sep 26, 1988.
6. Beverly Woolf, "Intelligent Tutoring Systems: A Survey," *Exploring Artificial Intelligence: Survey Talks from the National Conferences on the Artificial Intelligence*, p. 2, Morgan Kaufmann Publishers, Inc., 1988.
7. Olen D. Porter, *A Decision Support System for the Diagnosis of Aircraft Emergencies*, M.S. Thesis, Naval Postgraduate School, Monterey, California, Dec 1986.
8. Neil C. Rowe, *Artificial Intelligence Through Prolog*, Prentice-Hall Inc., 1988.
9. Thomas D. Garvey, "Expert Systems Research Focuses On Combat Emergency Procedures," *Aviation Week & Space Technology*, Oct 28, 1985.
10. Mike E. Harris, "McDonnell Douglas Flight Tests AI Maintenance Data Processor," *Aviation Week & Space Technology*, Feb 17, 1986.
11. Newell, A. and Simon, H., "Empirical Exploration with the Logic Theory Machine: A case study in heuristics," in Feigenbaum, E.A., and Feldman, J., eds, *Computers and Thought*, New York, McGraw-Hill, 1963.
12. "GPS : A Program that Simulates Human Thought," in Feigenbaum, E.A., and Feldman, J., eds, *Computers and Thought*, New York, McGraw-Hill, 1963.
13. Neil C. Rowe, "Means-ends Tutoring", *Technical Report NPS 52-90-003*, 1990.

14. Naval Air Systems Command, *NATOPS Flight Manual Navy Model RF-4B aircraft*, Jul 15, 1979.
15. Fernando, Pereira, C-Prolog User's Manual, *Technical Report, SRI International*, Mar 31, 1990.

## INITIAL DISTRIBUTION LIST

- |    |  |   |
|----|--|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22304-6145                                       | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, CA 93943-5002   | 2 |
| 3. | Department Chairman, Code 52<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943-5000     | 2 |
| 4. | Curriculum Office, Code 37<br>Computer Technology<br>Naval Postgraduate School<br>Monterey, CA 93943-5000                  | 2 |
| 5. | Professor Neil C. Rowe, Code 52Rp<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943     | 5 |
| 6. | Professor Yuh-jeng Lee, Code 52Le<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943     | 1 |
| 7. | Professor Robert B. McGhee, Code 52Mz<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| 8. | Professor Se-Hung Kwak, Code 52Kw<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, CA 93943     | 1 |

- |     |  |   |
|-----|--|---|
| 9.  | Major Young-Goo Hwang<br>SMC 2120<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| 10. | Captain Jae-Doo Jung<br>SMC 1504<br>Naval Postgraduate School<br>Monterey, CA 93943  | 1 |
| 11. | Captain Hung-Taek Kim<br>SMC 1930<br>Naval Postgraduate School<br>Monterey, CA93943  | 1 |
| 12. | Major Hee-Man Kwon<br>SMC 1375<br>Naval Postgraduate School<br>Monterey, CA93943     | 1 |
| 13. | Major Hong-Eun Kim<br>SMC 2665<br>Naval Postgraduate School<br>Monterey, CA 93943    | 1 |
| 14. | Major Gang-Yoon Lee<br>SMC 2186<br>Naval Postgraduate School<br>Monterey, CA 93943   | 1 |
| 15. | Major Chang-Ho Kim<br>SMC 1671<br>Naval Postgraduate School<br>Monterey, CA 93943    | 1 |
| 16. | Major Moungh-Hung Kang<br>54-1 Woo-Am Dong<br>Cheong-Ju City, Choong-Buk<br>KOREA    | 5 |

617-586